# Database Basics MS SQL Exam – 24 Apr 2017

Exam problems for the [“Database Basics” course @ SoftUni](https://softuni.bg/courses/databases-basics-ms-sql-server). Submit your solutions in the SoftUni judge system at <https://judge.softuni.bg/Contests/543/>.

**Database Fundamentals MSSQL – Washing Machine Service**

*Your task is to implement Washing Machine Service database based on the specs given below. Database will include information about broken machines which will be distributed to mechanics through so called “jobs”. Each mechanic may have several jobs and for each job he/she can require some parts. To let the mechanic get whats needed he/she has to make order for a particular part also specifying how many of it will be needed (specifying quantity).*

# Section 1. DDL (30 pts)

You have been given the E/R Diagram of the washing machine service:



Crate a database called **WMS**. You need to create **9 tables**:

* **Clients** – contains information about the customers that use the service
* **Mechanics** – contains information about employees
* **Jobs** – contains information about all machines that clients submitted for repairs
* **Models** – list of all washing machine models that the servie operates with
* **Orders** – contains information about orders for parts
* **Parts** – list of all parts the service operates with
* **OrderParts** – mapping table between Orders and Parts with additional Quantity field
* **PartsNeeded** – mapping table between Jobs and Parts with additional Quantity field
* **Vendors** – list of vendors that supply parts to the service

Include the following fields in each table. Unless otherwise specified, **all fields are required**.

**Clients**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| ClientId | 32-bit Integer | Primary table identificator, Identity |
| FirstName | String up to 50 symbols, ASCII | NOT NULL |
| LastName | String up to 50 symbols, ASCII | NOT NULL |
| Phone | String containing 12 symbols | String length is **exactly** 12 chars long, NOT NULL |

**Mechanics**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| MechanicId | 32-bit Integer | Primary table identificator, Identity |
| FirstName | String up to 50 symbols, ASCII | NOT NULL |
| LastName | String up to 50 symbols, ASCII | NOT NULL |
| Address | String up to 255 symbols, ASCII | NOT NULL |

**Jobs**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| JobId | 32-bit Integer | Primary table identificator, Identity |
| ModelId | 32-bit Integer | Relationship with table Models NOT NULL |
| Status | String up to 11 symbols, ASCII | Allowed values: 'Pending', 'In Progress' and 'Finished'; Default value is 'Pending' NOT NULL |
| ClientId | 32-bit Integer | Relationship with table Clients NOT NULL |
| MechanicId | 32-bit Integer | Relationship with table Mechanics; Can be NULL |
| IssueDate | Date | NOT NULL |
| FinishDate | Date | Can be NULL |

**Models**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| ModelId | 32-bit Integer | Primary table identificator, Identity |
| Name | String up to 50 symbols, ASCII | Unique NOT NULL |

**Orders**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| OrderId | 32-bit Integer | Primary table identificator, Identity |
| JobId | 32-bit Integer | Relationship with table Jobs NOT NULL |
| IssueDate | Date | Can be NULL |
| Delivered | Boolean | Default value is False NOT NULL |

**Parts**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| PartId | 32-bit Integer | Primary table identificator, Identity |
| SerialNumber | String up to 50 symbols, ASCII | Unique NOT NULL |
| Description | String up to 255 symbols, ASCII | Can be NULL |
| Price | Monetary value up to 9999.99 | Cannot be zero or negative NOT NULL |
| VendorId | 32-bit Integer | Relationship with table Vendors NOT NULL |
| StockQty | 32-bit Integer | Cannot be negative; Default value is 0 NOT NULL |

**OrderParts**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| OrderId | 32-bit Integer | Relationship with table Orders;  Primary table identificator |
| PartId | 32-bit Integer | Relationship with table Parts;  Primary table identificator |
| Quantity | 32-bit Integer | Cannot be zero or negative; Default value is 1 NOT NULL |

**PartsNeeded**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| JobId | 32-bit Integer | Relationship with table Jobs;  Primary table identificator |
| PartId | 32-bit Integer | Relationship with table Parts;  Primary table identificator |
| Quantity | 32-bit Integer | Cannot be zero or negative; Default value is 1 NOT NULL |

**Vendors**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| VendorId | 32-bit Integer | Primary table identificator, Identity |
| Name | String up to 50 symbols, ASCII | Unique NOT NULL |

## Database design

Submit all of your create statements to Judge. **Do not include** database creation statements.

**Look for hints in the details of your submission!**

**Solution:**

CREATE TABLE Clients

(

ClientId INT PRIMARY KEY IDENTITY,

FirstName NVARCHAR(50) NOT NULL,

LastName NVARCHAR(50) NOT NULL,

Phone CHAR(12) NOT NULL

)

CREATE TABLE Mechanics

(

MechanicId INT PRIMARY KEY IDENTITY,

FirstName NVARCHAR(50) NOT NULL,

LastName NVARCHAR(50) NOT NULL,

[Address] NVARCHAR(255) NOT NULL

)

CREATE TABLE Models

(

ModelId INT PRIMARY KEY IDENTITY,

[Name] NVARCHAR(50) UNIQUE NOT NULL

)

CREATE TABLE Jobs

(

JobId INT PRIMARY KEY IDENTITY,

ModelId INT FOREIGN KEY REFERENCES Models(ModelId) NOT NULL,

[Status] NVARCHAR(11) DEFAULT('Pending')

CHECK ([Status] = 'In Progress' OR [Status] = 'Finished' OR [Status] = 'Pending'),

ClientId INT FOREIGN KEY REFERENCES Clients(ClientId) NOT NULL,

MechanicId INT FOREIGN KEY REFERENCES Mechanics(MechanicId),

IssueDate DATE NOT NULL,

FinishDate DATE

)

CREATE TABLE Orders

(

OrderId INT PRIMARY KEY IDENTITY,

JobId INT FOREIGN KEY REFERENCES Jobs(JobId) NOT NULL,

IssueDate DATE,

Delivered BIT DEFAULT 0 NOT NULL

)

CREATE TABLE Vendors

(

VendorId INT PRIMARY KEY IDENTITY,

[Name] NVARCHAR(50) UNIQUE NOT NULL

)

CREATE TABLE Parts

(

PartId INT PRIMARY KEY IDENTITY,

SerialNumber NVARCHAR(50) UNIQUE NOT NULL,

[Description] NVARCHAR(255),

Price DECIMAL(6,2) NOT NULL CHECK (Price > 0),

VendorId INT FOREIGN KEY REFERENCES Vendors(VendorId),

StockQty INT DEFAULT 0 NOT NULL CHECK (StockQty > =0 )

)

CREATE TABLE OrderParts

(

OrderId INT NOT NULL,

PartId INT NOT NULL,

CONSTRAINT PK\_OrdersParts

PRIMARY KEY(OrderId, PartId),

CONSTRAINT FK\_OrdersParts\_Orders

FOREIGN KEY(OrderId)

REFERENCES Orders(OrderId),

CONSTRAINT FK\_OrdersParts\_Parts

FOREIGN KEY(PartId)

REFERENCES Parts(PartId),

Quantity INT DEFAULT 1 CHECK (Quantity > 0) NOT NULL

)

CREATE TABLE PartsNeeded

(

JobId INT NOT NULL,

PartId INT NOT NULL,

CONSTRAINT PK\_JobsParts

PRIMARY KEY(JobId, PartId),

CONSTRAINT FK\_JobsParts\_Jobs

FOREIGN KEY(JobId)

REFERENCES Jobs(JobId),

CONSTRAINT FK\_JobsParts\_Parts

FOREIGN KEY(PartId)

REFERENCES Parts(PartId),

Quantity INT DEFAULT 1 CHECK (Quantity > 0) NOT NULL

)

# Section 2. DML (10 pts)

**Before you start you have to import Data.sql. If you have created the structure correctly the data should be successfully inserted.**

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

## Insert

Let’s **insert** some sample data into the database. Write a query to add the following records into the corresponding tables. All Id’s should be auto-generated. Replace names that relate to other tables with the appropriate ID (look them up manually, there is no need to perform table joins).

**Clients**

|  |  |  |
| --- | --- | --- |
| **First Name** | **Last Name** | **Phone** |
| Teri | Ennaco | 570-889-5187 |
| Merlyn | Lawler | 201-588-7810 |
| Georgene | Montezuma | 925-615-5185 |
| Jettie | Mconnell | 908-802-3564 |
| Lemuel | Latzke | 631-748-6479 |
| Melodie | Knipp | 805-690-1682 |
| Candida | Corbley | 908-275-8357 |

**Parts**

|  |  |  |  |
| --- | --- | --- | --- |
| **Serial Number** | **Description** | **Price** | **Vendor Name** |
| WP8182119 | Door Boot Seal | 117.86 | Suzhou Precision Products |
| W10780048 | Suspension Rod | 42.81 | Shenzhen Ltd. |
| W10841140 | Silicone Adhesive | 6.77 | Fenghua Import Export |
| WPY055980 | High Temperature Adhesive | 13.94 | Qingdao Technology |

**Solution:**

INSERT INTO Clients(FirstName, LastName, Phone)

VALUES

('Teri', 'Ennaco', '570-889-5187'),

('Merlyn', 'Lawler', '201-588-7810'),

('Georgene', 'Montezuma', '925-615-5185'),

('Jettie', 'Mconnell', '908-802-3564'),

('Lemuel', 'Latzke', '631-748-6479'),

('Melodie', 'Knipp', '805-690-1682'),

('Candida', 'Corbley', '908-275-8357')

INSERT INTO Parts(SerialNumber, [Description], Price, VendorId)

VALUES

('WP8182119', 'Door Boot Seal', 117.86, 2),

('W10780048', 'Suspension Rod', 42.81, 1),

('W10841140', 'Silicone Adhesive', 6.77, 4),

('WPY055980', 'High Temperature Adhesive', 13.94, 3)

## Update

Assign all **Pending** jobs to the mechanic **Ryan Harnos** (look up his ID manually, there is no need to use table joins) and change their **status** to '**In Progress**'

**Solution:**

UPDATE Jobs

SET [Status] = 'In Progress', MechanicId = 3

WHERE Status = 'Pending'

## Delete

Cancel Order with ID 19 – delete the order from the database and all associated entries from the mapping table.

**Solution:**

DELETE FROM OrderParts

WHERE OrderId = 19

DELETE FROM Orders

WHERE OrderId = 19

# Section 3. Querying (45 pts)

**You need to start with a fresh dataset, so run the Data.sql script again. It includes a section that will delete all records and replace them with the starting set, so you don’t need to drop your database.**

## Clients by Name

Select all **clients** ordered by **last name** (ascending) then by **client ID** (ascending).

Required columns:

* First Name
* Last Name
* Phone

Example:

|  |  |  |
| --- | --- | --- |
| **FirstName** | **LastName** | **Phone** |
| Tasia | Andreason | 201-920-9002 |
| Thaddeus | Ankeny | 916-920-3571 |
| Loren | Asar | 570-648-3035 |
| … | … | … |

**Solution:**

SELECT FirstName, LastName, Phone FROM Clients

ORDER BY LastName, ClientId

## Job Status

Find all active **jobs** (that aren’t **Finished**) and display their **status** and **issue date**. Order by issue date and by job ID (both ascending).

Required columns:

* Status
* Issue Date

Example:

|  |  |
| --- | --- |
| **Status** | **IssueDate** |
| In Progress | 2017-04-06 |
| In Progress | 2017-04-10 |
| In Progress | 2017-04-12 |
| … | … |

**Solution:**

SELECT Status, IssueDate FROM Jobs

WHERE Status <> 'Finished'

ORDER BY IssueDate, JobId

## Mechanic Assignments

Select all **mechanics** with their **jobs**. Include job **status** and **issue date**. Order by mechanic Id, issue date, job Id (all ascending).

Required columns:

* Mechanic Full Name
* Job Status
* Job Issue Date

Example:

|  |  |  |
| --- | --- | --- |
| **Mechanic** | **Status** | **IssueDate** |
| Joni Breland | Finished | 2017-01-12 |
| Joni Breland | Finished | 2017-01-17 |
| Joni Breland | Finished | 2017-01-24 |
| … | … | … |

**Solution:**

SELECT CONCAT(m.FirstName, ' ', m.LastName) AS Mechanic,

j.[Status],

j.IssueDate

FROM Jobs AS j

JOIN Mechanics AS m ON j.MechanicId = m.MechanicId

ORDER BY m.MechanicId, j.IssueDate, j.JobId

## Current Clients

Select the names of all **clients** with active **jobs** (not **Finished**). Include the **status** of the job and **how many days** it’s been since it was submitted. Assume the **current date** is 24 April 2017. Order results by time length (descending) and by client ID (ascending).

Required columns:

* Client Full Name
* Days going – how many days have passed since the issuing
* Status

**Solution:**

SELECT CONCAT(c.FirstName, ' ', c.LastName) AS Client,

DATEDIFF(DAY, j.IssueDate, '2017-04-24') AS [Days going],

j.[Status]

FROM Clients AS c

JOIN Jobs AS j ON j.ClientId = c.ClientId

WHERE j.[Status] <> 'Finished'

ORDER BY DATEDIFF(DAY, j.IssueDate, '2017-04-24') DESC, c.ClientId

Example:

|  |  |  |
| --- | --- | --- |
| **Client** | **Days going** | **Status** |
| Gertude Witten | 18 | In Progress |
| Brittni Gillaspie | 14 | In Progress |
| Levi Munis | 12 | In Progress |
| … | … | … |

## Mechanic Performance

Select all **mechanics** and the **average time** they take to finish their assigned **jobs**. Calculate the average as an integer. Order results by mechanic ID (ascending).

Required columns:

* Mechanic Full Name
* Average Days – average number of days the machanic took to finish the job

**Solution:**

SELECT CONCAT(m.FirstName, ' ', m.LastName) AS Mechanic,

CAST(AVG(DATEDIFF(DAY, j.IssueDate, j.FinishDate)) AS INT) AS [Average days]

FROM Mechanics AS m

JOIN Jobs AS j ON j.MechanicId = m.MechanicId

WHERE j.FinishDate IS NOT NULL

GROUP BY CONCAT(m.FirstName, ' ', m.LastName), m.MechanicId

ORDER BY m.MechanicId

Example:

|  |  |
| --- | --- |
| **Mechanic** | **Average Days** |
| Joni Breland | 9 |
| Malcolm Tromblay | 10 |
| Ryan Harnos | 5 |
| … | … |

## Hard Earners

Select the first 3 **mechanics** who have more than 1 active **job** (not **Finished**). Order them by number of jobs (descending) and by mechanic ID (ascending).

Required columns:

* Mechanic Full Name
* Number of Jobs

**Solution:**

SELECT CONCAT(m.FirstName, ' ', m.LastName) AS Mechanic,

COUNT(j.JobId) AS Jobs

FROM Mechanics AS m

JOIN Jobs AS j ON j.MechanicId = m.MechanicId

WHERE j.Status <> 'Finished'

GROUP BY CONCAT(m.FirstName, ' ', m.LastName) , m.MechanicId

HAVING COUNT(j.JobId) > 1

ORDER BY COUNT(j.JobId) DESC, m.MechanicId

Example:

|  |  |
| --- | --- |
| **Mechanic** | **Jobs** |
| Nickolas Juvera | 3 |
| Malcolm Tromblay | 2 |

## Parts Cost

Display the **total cost** of all **parts** ordered during the last three weeks. Assume the **current date** is 24 April 2017.

Required columns:

* Parts Total Cost

**Solution:**

SELECT SUM( [Parts Total] ) AS [Parts Total]

FROM (

SELECT SUM(p.Price \* op.Quantity) AS [Parts Total]

FROM Parts AS p

JOIN OrderParts AS op ON op.PartId = p.PartId

JOIN Orders AS o ON o.OrderId = op.OrderId

WHERE DATEDIFF(WEEK, o.IssueDate, '2017-04-27') <= 3

GROUP BY P.PartId) AS f

Example:

|  |
| --- |
| **Parts Total** |
| 109.62 |

## Past Expenses

Select all **finished jobs** and the total cost of all **parts** that were **ordered** for them. Sort by total cost of parts ordered (descending) and by job ID (ascending).

Required columns:

* Job ID
* Total Parts Cost

Example:

|  |  |
| --- | --- |
| **JobId** | **Total** |
| 17 | 173.60 |
| 12 | 140.50 |
| 1 | 91.86 |
| … | … |

**Solution:**

SELECT j.JobId, IIF((SUM(op.Quantity \* p.Price) IS NULL), 0, SUM(op.Quantity \* p.Price) )AS Total FROM Jobs AS j

LEFT JOIN Orders AS o ON o.JobId = j.JobId

LEFT JOIN OrderParts AS op ON op.OrderId = o.OrderId

LEFT JOIN Parts AS p ON p.PartId = op.PartId

WHERE j.[Status] = 'Finished'

GROUP BY j.JobId

ORDER BY SUM(op.Quantity \* p.Price) DESC, j.JobId

## Model Repair Time

Select all **models** with the **average time** it took to service, out of all the times it was repaired. Calculate the average as an integer value. Order the results by average service time ascending.

Required columns:

* Model ID
* Name
* Average Service Time – average number of days it took to finish the job; **note the word 'days' attached at the end!**

**Solution:**

SELECT f.ModelId, f.[Name], CONCAT(f.[Average Service Time], ' days') FROM (SELECT m.ModelId, m.[Name], AVG(DATEDIFF(DAY, j.IssueDate, j.FinishDate)) AS [Average Service Time] FROM Models AS m

JOIN Jobs AS j ON j.ModelId = m.ModelId

WHERE j.[Status] = 'Finished'

GROUP BY m.ModelId, m.[Name] ) AS f

ORDER BY f.[Average Service Time]

Example:

|  |  |  |
| --- | --- | --- |
| **ModelId** | **Name** | **Average Service Time** |
| 5 | Maelstrom L700 | 6 days |
| 4 | LN 100F | 7 days |
| 1 | Maelstrom L300 | 8 days |
| … | … | … |

## Faultiest Model

Find the **model** that breaks the most (has the highest number of **jobs** associated with it). Include the cost of **parts** **ordered** for it. If there are more than one models that were serviced the same number of times, **list them all**.

Required columns:

* Name
* Times Serviced – number of assiciated jobs
* Parts Total – cost of all parts ordered for the jobs

**Solution:**

SELECT m.[Name],

COUNT(j.JobId) AS [Times Serviced],

SUM(op.Quantity \* p.Price) AS [Parts Total]

FROM Jobs AS j

JOIN Models AS m ON j.ModelId = m.ModelId

LEFT JOIN Orders AS o ON j.JobId = o.JobId

LEFT JOIN OrderParts AS op ON op.OrderId = o.OrderId

LEFT JOIN Parts AS p ON p.PartId = op.PartId

WHERE j.[Status] = 'Finished'

GROUP BY m.[Name]

ORDER BY COUNT(j.JobId) DESC

Example:

|  |  |  |
| --- | --- | --- |
| **Model** | **Times Serviced** | **Parts Total** |
| Neko GG | 14 | 321.72 |

# Section 4. Programmability (15 pts)

## Cost of Order

Create a **user defined function** (udf\_GetCost) that receives a **job’s ID** and returns the total **cost** of all **parts** that were **ordered** for it. Return 0 if there are **no orders**.

**Solution:**

CREATE FUNCTION udf\_GetCost(@jobId INT)

RETURNS DECIMAL(18,2)

AS

BEGIN

DECLARE @partsPrice DECIMAL(18,2)

SET @partsPrice = (SELECT SUM(p.Price \* pn.Quantity) FROM Jobs AS j

JOIN PartsNeeded AS pn ON pn.JobId = j.JobId

JOIN Parts AS p ON pn.PartId = p.PartId

WHERE j.JobId = @jobId

GROUP BY j.JobId)

IF(@partsPrice IS NULL)

BEGIN

RETURN 0

END

RETURN @partsPrice

END

Parameters:

* JobId

Example usage:

|  |  |
| --- | --- |
| **Query** | |
| SELECT dbo.udf\_GetCost(1) | |
| Id | Result |
| 1 | 91.86 |
| 3 | 40.97 |
| … | … |