Database Basics MS SQL Exam – 24 Apr 2017

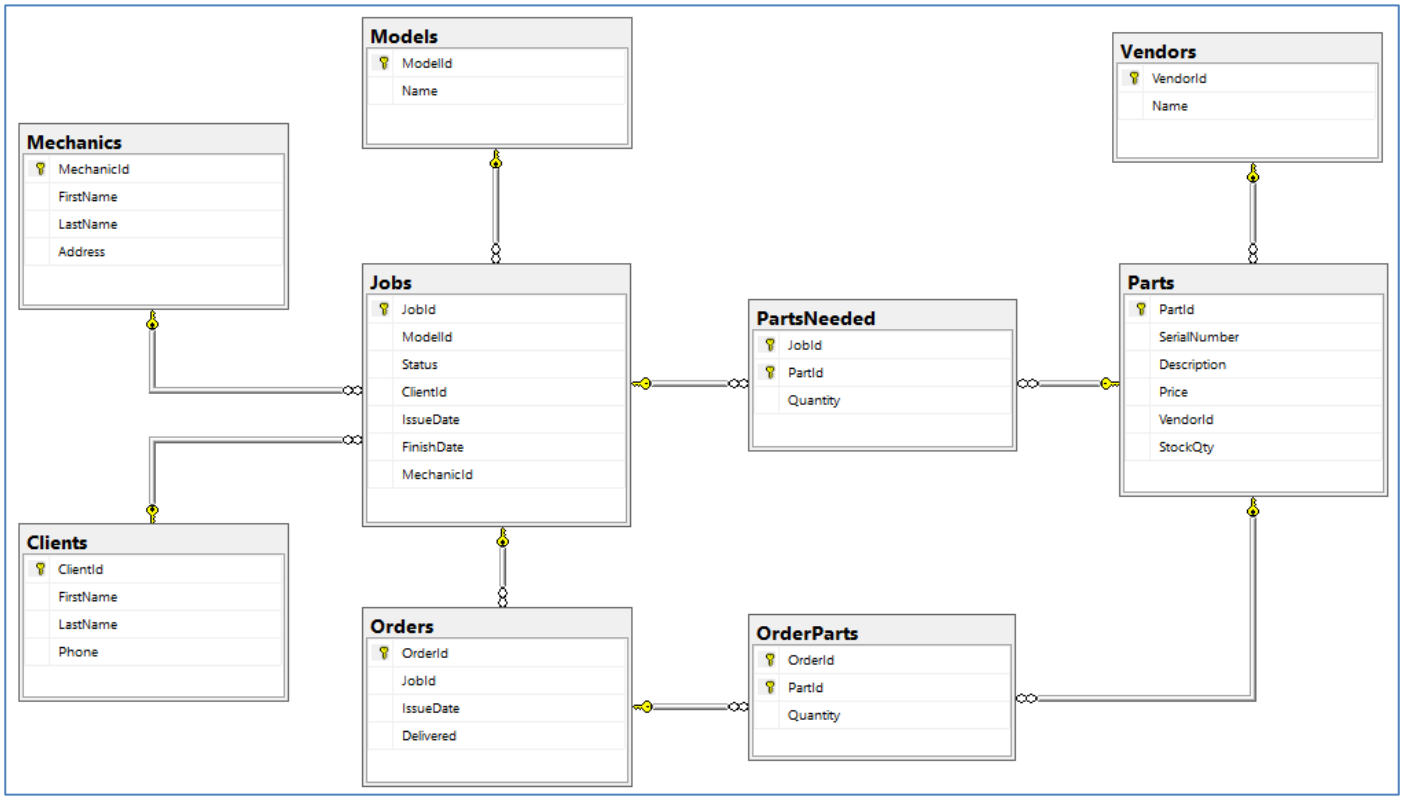
Exam problems for the "Database Basics" course @ SoftUni. Submit your solutions in the SoftUni judge system at https://judge.softuni.bg/Contests/543/.

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	
LastName	String up to 50 symbols, ASCII	
Phone	String containing 12 symbols	String length is **exactly** 12 chars long

Mechanics

Column Name	Data Type	Constraints
MechanicId	32-bit Integer	Primary table identificator, Identity
FirstName	String up to 50 symbols, ASCII	
LastName	String up to 50 symbols, ASCII	
Address	String up to 255 symbols, ASCII	

Jobs

Column Name	Data Type	Constraints
JobId	32-bit Integer	Primary table identificator, Identity
ModelId	32-bit Integer	Relationship with table Models
Status	String up to 11 symbols, ASCII	Allowed values: 'Pending', 'In Progress' and 'Finished'; Default value is 'Pending'
ClientId	32-bit Integer	Relationship with table Clients
MechanicId	32-bit Integer	Relationship with table Mechanics; Can be **NULL**
IssueDate	Date	
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

Orders

Column Name	Data Type	Constraints
OrderId	32-bit Integer	Primary table identificator, Identity
JobId	32-bit Integer	Relationship with table Jobs
IssueDate	Date	Can be **`NULL`**
Delivered	Boolean	Default value is **`False`**

Parts

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

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

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

Vendors

Column Name	Data Type	Constraints
VendorId	32-bit Integer	Primary table identificator, Identity
Name	String up to 50 symbols, ASCII	Unique

1. 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!

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:

2. 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

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**'.

4. Delete

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

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.

5. 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
…	…	…

6. 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
…	…

7. 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
…	…	…

8. 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

Example:

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

9. 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

Example:

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

10. 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

Example:

Mechanic	Jobs
Nickolas Juvera	3
Malcolm Tromblay	2

11. Available Mechanics

Select all **mechanics** without active **jobs** (include mechanics which don't have any job assigned or all of their jobs are finished). Order by ID (ascending).

Required columns:

- Mechanic Full Name

Example:

Available
Joni Breland
Ryan Harnos
…

12. 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

Example:

Parts Total
109.62

13. 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
…	…

14. 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!**

Example:

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

15. 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

Example:

Model	Times Serviced	Parts Total
Neko GG	14	321.72

16. Missing Parts

List all **parts** that are **needed** for active **jobs** (not **Finished**) without sufficient quantity **in stock** and in pending **orders** (the sum of parts in stock and parts ordered is less than the required quantity). Order them by part ID (ascending).

Required columns:

- Part ID
- Description
- Required – number of parts required for active jobs
- In Stock – how many of the part are currently in stock
- Ordered – how many of the parts are expected to be delivered (associated with order that is not Delivered)

Example:

PartId	Description	Required	In Stock	Ordered
12	Shock Dampener	2	1	0
14	Door Handle	1	0	0
17	Lid Switch Assembly	1	0	0

Section 4. Programmability (15 pts)

17. 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**.

Parameters:

- JobId

Example usage:

Query	
`SELECT dbo.udf_GetCost(1)`	
Id	**Result**
1	91.86
3	40.97
…	…

18. Place Order

Your task is to create a user defined procedure (**usp_PlaceOrder**) which accepts **job ID**, part **serial number** and **quantity** and creates an **order** with the specified **parameters**. If an order **already exists** for the given job **that and the order is not issued** (order's issue date is **NULL**), add the new product to it. If the part is already listed in the order, add the quantity to the existing one.

When a new order is created, set it's IssueDate to **NULL**.

Limitations:

- An order **cannot** be placed for a job that is **Finished**; error message ID 50011 "**This job is not active!**"
- The quantity **cannot** be zero or negative; error message ID 50012 "**Part quantity must be more than zero!**"
- The job with given ID **must exist** in the database; error message ID 50013 "**Job not found!**"
- The part with given serial number **must exist** in the database ID 50014 "**Part not found!**"

If any of the requirements aren't met, rollback any changes to the database you've made and **throw an exception** with the **appropriate message** and **state 1**.

Parameters:

- JobId
- Part Serial Number
- Quantity

Example usage:

Query

`DECLARE @err_msg AS NVARCHAR(MAX);`
`BEGIN TRY`
`  EXEC usp_PlaceOrder 1, 'ZeroQuantity', 0`
`END TRY`
 	
`BEGIN CATCH`
`  SET @err_msg = ERROR_MESSAGE();`
`  SELECT @err_msg`
`END CATCH`

Response
Part quantity must be more than zero!

19. Detect Delivery

Create a **trigger** that detects when an order's **delivery status** is changed from **`False`** to **`True`** which adds the quantities of all ordered parts to their stock quantity value (Qty).

Example usage:

Query

`UPDATE Orders`
`SET Delivered = 1`
`WHERE OrderId = 21`

Response
(1 row(s) affected)

Section 5. Bonus (10 pts)

20. Vendor Preference

List all mechanics and their preference for each vendor as a percentage of parts' quantities they ordered for their jobs. Express the percentage as an integer value. Order them by mechanic's full name (ascending), number of parts from each vendor (descending) and by vendor name (ascending).

Required columns:

- Mechanic Full Name
- Vendor Name
- Parts ordered from vendor

- Preference for Vendor (percantage of parts out of all parts count ordered by the mechanic)

Example:

Mechanic	Vendor	Parts	Preference
Gary Nunlee	Shenzhen Ltd.	2	100%
Jess Chaffins	Qingdao Technology	4	57%
Jess Chaffins	Suzhou Precision Products	2	28%
Jess Chaffins	Fenghua Import Export	1	14%
…	…	…	…