Database Basics MS SQL Exam – 24 Jun 2018

Exam problems for the "Database Basics" course @ SoftUni.

Submit your solutions in the SoftUni Judge system at https://judge.softuni.bg/

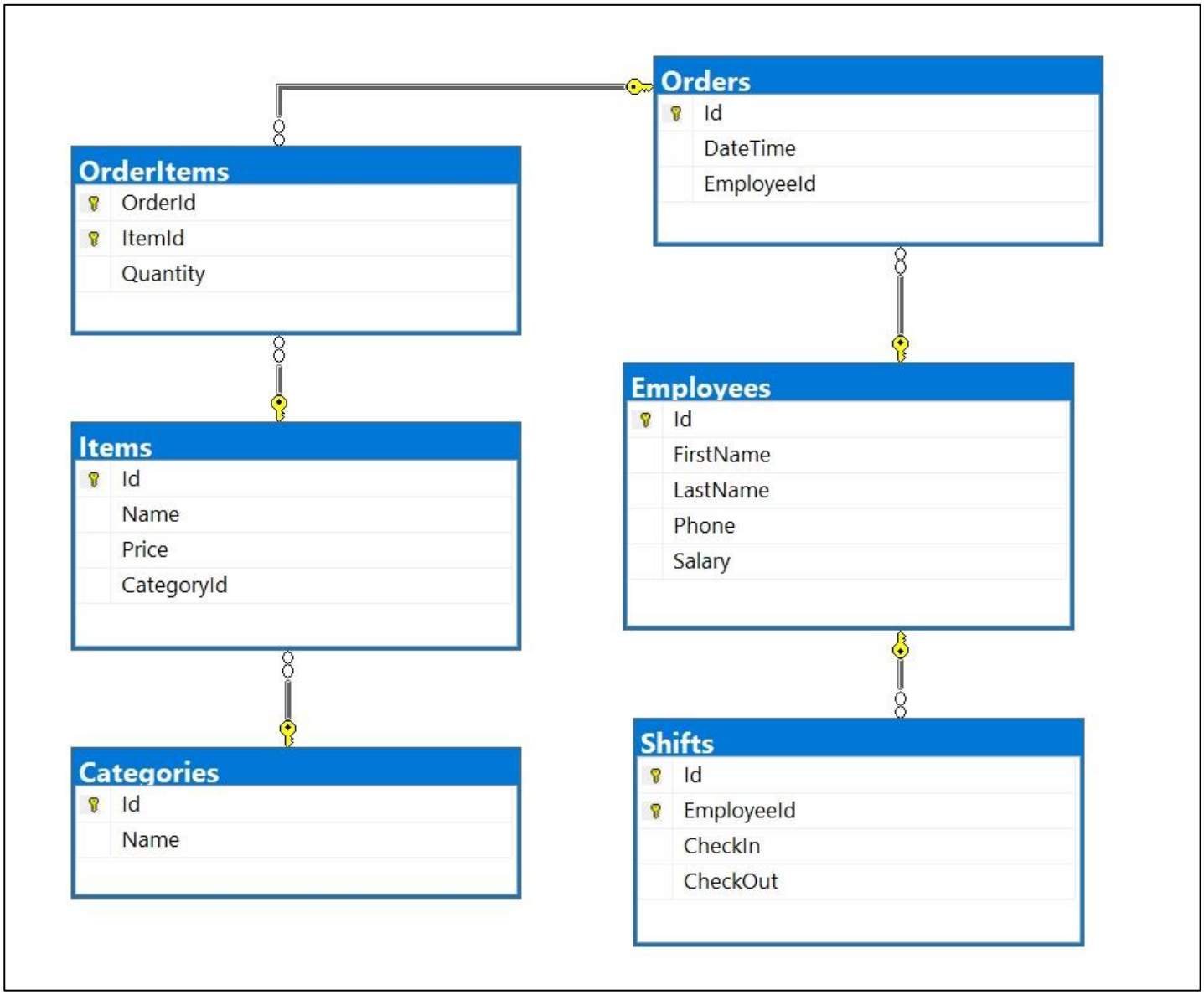
Supermarket

After feeling extremely disappointed with your previous job at "**Krivodol Trip Service LLC**", you have now started working for a new and much better company – "**Pustinqk Software**". From the very beginning your new boss saw a huge potential in you and has assigned you a very exciting project. In **6 hours**, you must develop a complicated system for a small shop, which has now grown bigger.

Your database must contain information about the **employees** and their **work hours**. You must also include information about the **products** and their **orders**.

Section 1. DDL (30 pts)

You are given an E/R Diagram of the Trip Service:



Create a database called **Supermarket**. You need to create **6 tables**:

- **Categories** – contains information about the **item categories**.
- **`Items`** – contains information about the items and their categories.
- **`Orders`** – contains information about all of the store orders.
- **`OrderItems`** – contains information about every order's items.
- **`Employees`** – contains information about the employees.
- **`Shifts`** – contains information about **check-in** tracking for **employees**.

Categories

Column	Data Type	Constraints
Id	**Integer** from **0** to **2,147,483,647**	Unique table **identificator**, **Identity**
Name	**String** up to 30 symbols, Unicode	**NULL** is **not** allowed

Items

Column	Data Type	Constraints
Id	**Integer** from **0** to **2,147,483,647**	Unique table **identificator**, **Identity**
Name	**String** up to 30 symbols, Unicode	**NULL** is **not** allowed
Price	**Decimal** number with **two-digit** precision	**NULL** is **not** allowed
CategoryId	**Integer** from **0** to **2,147,483,647**	**NULL** is **not** allowed, Relationship with table Categories

Employees

Column	Data Type	Constraints
Id	**Integer** from **0** to **2,147,483,647**	Unique table **identificator**, **Identity**
FirstName	**String** up to 50 symbols, Unicode	**NULL** is **not** allowed
LastName	**String** up to 50 symbols, Unicode	**NULL** is **not** allowed
Phone	**String** with **exactly 12** symbols	**NULL** is **not** allowed
Salary	**Decimal** number with **two-digit** precision	**NULL** is **not** allowed

Orders

Column	Data Type	Constraints
Id	**Integer** from **0** to **2,147,483,647**	Unique table **identificator**, **Identity**
DateTime	**DateTime**	**NULL** is **not** allowed
EmployeeId	**Integer** from **0** to **2,147,483,647**	**NULL** is **not** allowed, Relationship with table Employees

OrderItems

Column	Data Type	Constraints
OrderId	**Integer** from **0** to **2,147,483,647**	**NULL** is **not** allowed, Relationship with table **Orders**
ItemId	**Integer** from **0** to **2,147,483,647**	**NULL** is **not** allowed, Relationship with table **Items**
Quantity	**Integer** from **0** to **2,147,483,647**	**NULL** is **not** allowed, must be **at least 1**

Shifts

Column	Data Type	Constraints
Id	**Integer** from **0** to **2,147,483,647**	Unique table **identificator**, **Identity**
EmployeeId	**Integer** from **0** to **2,147,483,647**	Unique table **identificator** , Relationship with table **Employees**
CheckIn	**DateTime**	**NULL** is **not** allowed
CheckOut	**DateTime**	**NULL** is **not** allowed, must be after CheckIn date

1. Database Design

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

Section 2. DML (10 pts)

Before you start, you must import "`DataSet-Supermarket.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:

2. Insert

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

Employees

FirstName	LastName	Phone	Salary
Stoyan	Petrov	888-785-8573	500.25
Stamat	Nikolov	789-613-1122	999995.25
Evgeni	Petkov	645-369-9517	1234.51
Krasimir	Vidolov	321-471-9982	50.25

Items

Name	Price	CategoryId
Tesla battery	154.25	8
Chess	30.25	8
Juice	5.32	1
Glasses	10	8
Bottle of water	1	1

3. Update

Make all items' prices **27% more expensive** where the **category ID** is either **1**, **2** or **3.**

4. Delete

Delete all order items where the order id is 48 (be careful with the relationships)

Section 3. Querying (40 pts)

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

5. Richest People

Select all **employees** who have a **salary** above **6500**. Order them by **first name**, then by **employee id**.

Example

Id	FirstName
19	Arney
32	Arther
2	Celie
11	Emlynn
…	…

6. Cool Phone Numbers

Select all **full names** from employees, whose phone number start with '**3**'.

Order them by **first name (ascending)**, then by phone number **(ascending)**.

Example

Full Name	Phone Number
Adolphe Leacock	339-446-1263
Audie Risebarer	341-873-1275
Demeter Langdale	312-175-3209
Jordanna Asmus	323-785-5898
…	…

7. Employee Statistics

Select all **employees** who have orders with the total count of the orders they processed. Order them by their **orders count (descending)**, then by **first name**. Select their **first name**, **last name** and **total count** of **orders**.

Example

FirstName	LastName	Count
Bart	Jozwiak	123
Beverlee	Raveau	116
Ashley	Topliss	106
Gayler	Wike	103
Celie	De Cruce	96
…	…	…

8. Hard Workers Club

Select all **employees** whose workday is **over 7 hours long on average**, based on their **check in/check out times**. Select their **first, last name** and **average work hours**.

Order them by **work hours (descending),** then by **employee ID**.

Example

FirstName	LastName	Work hours

Gill	Wasiela	9
Celie	De Cruce	8
Jordanna	Asmus	8
Lucie	Dickinson	8
…	…	…

9. The Most Expensive Order

Find the most expensive order. Select its **id** and total item price. Consider the item **quantity** when calculating the price.

Example

OrderId	TotalPrice
479	14087.84

10. Rich Item, Poor Item

Find the top 10 **most expensive** and **cheapest** item in **each order.**

Order the results by **most expensive item's price** (**descending),** then by order id (**ascending**).

Example

OrderId	ExpensivePrice	CheapPrice
1	360.00	3.14
6	360.00	1.50
10	360.00	1.23
39	360.00	2.00
…	…	

11. Cashiers

Find all employees who have orders. Select their id, first name and last name. Order them by **employee id**.

Example

Id	First Name	Last Name
2	Celie	De Cruce
5	Lucie	Dickinson
8	Adaline	Gilogly
…	…	…

12. Lazy Employees

Find all employees, who have below 4 work hours per day.

Order them by employee id.

Example

Id	Full Name
1	Krishnah Lalor
4	Jasmine Forsdike
7	Ole De la Feld
…	…

13. Sellers

Find the top 10 employees with their full name, orders' total price and item count.

Count only orders which were **ordered before 2018-06-15**.

Order them by **total sum (descending)**, then by **item count (descending)**

Example

Full Name	Total Price	Items
Bart Jozwiak	37612.33	2497
Adaline Gilogly	26989.77	1765
Celie De Cruce	25692.80	1773
Gayler Wike	24754.87	2350
Lucie Dickinson	23707.26	1223
…	…	…

14. Tough days

Find all records of the employees who don't have orders and who work over 12 hours.

Select only their full name and day of the week.

Order the results by **employee id**.

Note: By the American Standards, Sunday is the first day of week.

Example

Full Name	Day of week
Krishnah Lalor	Sunday
Jordanna Asmus	Monday
Ole De la Feld	Friday
Ole De la Feld	Thursday
…	…

15. Top Order per Employee

Find all information of the employees who have orders. Select their full name, duration of the work day (**in hours**) and total price of all sold products. Find only the **top orders** (top orders with highest total price).

Sort them by **full name** (ascending), **work hours** (descending) and **total price** (descending)

Example

Full Name	WorkHours	TotalPrice

Adaline Gilogly	5	9460.00
Adolphe Leacock	5	14087.84
Anatola Lydon	8	4090.80
…	…	…

16. Average Profit per Day

Find the **average profit** for each day. Select the **day of month** and **average daily profit** of sold products.

Sort them by **day of month** (ascending) and format the profit to the **second digit** after the decimal point.

Example

Day	Total profit
1	254.79
3	211.49
4	115.89
5	83.26
6	111.47
7	101.49
8	140.65
10	90.17
11	281.59
12	162.31
13	127.65
…	…

17. Top Products

Find information about **all products**. Select their name, category, how many of them were sold and the total profit they produced.

Sort them by **total profit** (**descending**) and **their count** (**descending**)

Example

Item	Category	Count	TotalPrice
TV	Miscellaneous	308	110880.00
Tires	Miscellaneous	524	78600.00
Mattress	Miscellaneous	298	29800.00
Camera	Miscellaneous	352	28160.00
…	…	…	…

Section 4. Programmability (20 pts)

18. Promotion days

Create a **user defined function**, named **`udf_GetPromotedProducts(@CurrentDate, @StartDate, @EndDate, @Discount, @FirstItemId, @SecondItemId, @ThirdItemId)`**, that receives a **current date**, a **start date** for the promotion, an **end date** for the promotion, a **discount**, a **first item id**, a **second item id** and **third item id**.

The function should print the discounted price of the items, based on these conditions:

- The first, second and third items must exist in the database.
- The current date must be between the start date and end date.

If both conditions are true, you must discount the price and print the following message in the format:

- "**{FirstItemName} price: {@FirstItemPrice} <-> {SecondItemName} price: {@SecondItemPrice} <-> {ThirdItemName} price: {@ThirdItemPrice}**"

If one of the items is not in the database, the function should return "**One of the items does not exists!**"
If the current date is not between the start date and end date, the function should return "**The current date is not within the promotion dates!**"

Note: Do not update any records in the database!

Example:

Query
`SELECT dbo.udf_GetPromotedProducts('2018-08-02', '2018-08-01', '2018-08-03',13, 3,4,5)`
Output
`Water price: 0.74 <-> Juice price: 1.31 <-> Ayran price: 4.35`

Query
`SELECT dbo.udf_GetPromotedProducts('2018-08-01', '2018-08-02', '2018-08-03',13,3 ,4,5)`
Output
`The current date is not within the promotion dates!`

19. Cancel order

Create a **user defined stored procedure**, named **usp_CancelOrder(@OrderId, @CancelDate)**, that receives an **order id** and **date**, and attempts to **delete the current order**. An order will only be deleted if all of these conditions **pass**:

- If the **order** doesn't exists, then it **cannot be deleted. Raise an error** with the message "**The order does not exist!**"
- If the **cancel date** is 3 days after the issue date, **raise an error** with the message "**You cannot cancel the order!**"

If all the above conditions pass, **delete the order**.

Example usage:

Query	Output

`EXEC usp_CancelOrder 1, '2018-06-02'`
`SELECT COUNT(*) FROM Orders`
`SELECT COUNT(*) FROM OrderItems`
	`998`
	`2455`

`EXEC usp_CancelOrder 1, '2018-06-15'`	`You cannot cancel the order!`
`EXEC usp_CancelOrder 124231, '2018-06-15'`	`The order does not exist!`

20. Deleted Order

Create a new table **"DeletedOrders**" with columns **(OrderId, ItemId, ItemQuantity)**. Create a **trigger**, which fires when order is deleted. After deleting the order, **insert all of the data into the new table "DeletedOrders"**.

Note: Submit only your **CREATE TRIGGER** statement!

Example usage:

Query

`DELETE FROM OrderItems`
`WHERE OrderId = 5`
 	
`DELETE FROM Orders`
`WHERE Id = 5`

Response

`(5 rows affected)`
 	
`(5 rows affected)`
 	
`(1 rows affected)`