# Databases Sample Exam (Feb 2015)

Your exam consists of several parts, explained below. You may work independently on each exam part. Submit your solutions in the automated judge system: <http://judge.softuni.bg/Contests/70/Databases-Sample-Exam>.

## Part I – SQL Queries

You are given a **MS SQL Server database "Ads"** holding advertisements, organized by categories and towns, available as SQL script. Your task is to write SQL queries for displaying data from the given database.

### All Ad Titles +

Display all **ad titles in ascending order**. Submit for evaluation the result grid **with headers**.

|  |
| --- |
| **Title** |
| 2 bed apartment to let |
| … |

1 score

### Ads in Date Range +

Find **all ads** created between **26-December-2014** (00:00:00) and **1-January-2015** (23:59:59) sorted by date. Submit for evaluation the result grid with headers.

|  |  |
| --- | --- |
| **Title** | **Date** |
| Free cat | 2014-12-26 05:14:30.000 |
| Skandika Vancouver 6 Tent | 2014-12-26 11:42:40.000 |
| … | … |

2 score

### \* Ads with "Yes/No" Images -

Display all ad **titles** and **dates** along with a column named "**Has Image**" holding "**yes**" or "**no**" for all ads sorted by their Id. Submit for evaluation the result grid with headers.

|  |  |  |
| --- | --- | --- |
| **Title** | **Date** | **Has Image** |
| Just a Joke | 2014-12-24 23:53:17.000 | No |
| 2 bed apartment to let | 2014-12-24 20:21:11.000 | No |
| BMW 320 for Sale | 2014-12-26 12:37:50.000 | Yes |
| … | … | … |

4 score

### Ads without Town, Category or Image +

Find all ads that have no town or no category or no image sorted by Id. Show all their data. Submit for evaluation the result grid with headers.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Id** | **Title** | **Text** | **ImageDataURL** | **OwnerId** | **CategoryId** | **TownId** | **Date** | **StatusId** |
| 1 | Just a Joke | I sell my… | NULL | 956b34b3… | NULL | NULL | 2014-12-… | 3 |
| 2 | 2 bed … | Available… | NULL | 10216432-… | 3 | 5 | 2014-12-… | 3 |
| … | … | … | … | … | … | … | … | … |

2 score

### Ads with Their Town +

Find all **ads along with their towns** sorted by Id. Display the ad **title** and **town** (even when there is no town). Name the columns exactly like in the table below. Submit for evaluation the result grid with headers.

|  |  |
| --- | --- |
| **Title** | **Town** |
| Just a Joke | NULL |
| 2 bed apartment to let | Ruse |
| … | … |

2 score

### Ads with Their Category, Town and Status

Find all **ads along with their categories, towns and statuses** sorted by **Id**. Display the ad title, category name, town name and status. Include all ads without town or category or status. Name the columns exactly like in the table below. Submit for evaluation the result grid with headers.

|  |  |  |  |
| --- | --- | --- | --- |
| **Title** | **CategoryName** | **TownName** | **Status** |
| Just a Joke | NULL | NULL | Published |
| 2 bed apartment to let | Properties | Ruse | Published |
| … | … | … | … |

3 score

### Filtered Ads with Category, Town and Status

Find all **Published** ads **from Sofia, Blagoevgrad or Stara Zagora**, **along with their category, town and status** sorted by **title**. Display the ad title, category name, town name and status. Name the columns exactly like in the table below. Submit for evaluation the result grid with headers.

|  |  |  |  |
| --- | --- | --- | --- |
| **Title** | **CategoryName** | **TownName** | **Status** |
| BMW 320 for Sale | Cars | Sofia | Published |
| Free cat. Great hunter | Animals | Blagoevgrad | Published |
| … | … | … | … |

3 score

### Earliest and Latest Ads

Find the dates of the earliest and the latest ads. Name the columns exactly like in the table below. Submit for evaluation the result grid with headers.

|  |  |
| --- | --- |
| **MinDate** | **MaxDate** |
| 2014-12-23 19:04… | 2015-02-19 17:36… |

1 score

### Latest 10 Ads

Find the **latest 10 ads** sorted by date in descending order. Display for each ad its **title**, **date** and **status**. Name the columns exactly like in the table below. Submit for evaluation the result grid with headers.

|  |  |  |
| --- | --- | --- |
| **Title** | **Date** | **Status** |
| Lexus SC 430 Gold (2006) | 2015-02-19 17:36:00.000 | Inactive |
| 2012 Nissan Altima 3.5 SR 2dr Coupe CVT | 2015-02-02 22:44:00.000 | Waiting Approval |
| … | … | … |

2 score

### Not Published Ads from the First Month - !!!

Find all not published ads,created in the same month and year like the earliest ad. Display for each ad its **id**, **title**, **date** and **status**. Sort the results by Id. Name the columns exactly like in the table below. Submit for evaluation the result grid with headers.

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **Title** | **Date** | **Status** |
| 12 | Free cat | 2014-12-26 05:14:30.000 | Waiting Approval |
| … | … | … | … |

4 score

### Ads by Status

Display the **count of ads in each status**. Sort the results by **status**. Name the columns exactly like in the table below. Submit for evaluation the result grid with headers.

|  |  |
| --- | --- |
| **Status** | **Count** |
| Inactive | 3 |
| Published | 11 |
| … | … |

2 score

### Ads by Town and Status

Display the **count of ads for each town and each status**. Sort the results by **town**, then by **status**. Display only non-zero counts. Name the columns exactly like in the table below. Submit for evaluation the result grid with headers.

|  |  |  |
| --- | --- | --- |
| **Town Name** | **Status** | **Count** |
| Blagoevgrad | Inactive | 1 |
| Blagoevgrad | Published | 1 |
| Burgas | Published | 1 |
| Dobrich | Published | 1 |
| Dobrich | Waiting Approval | 2 |
| … | … | … |

3 score

### \* Ads by Users - !!!

Find the **count of ads for each user**. Display the **username**, **ads count** and "**yes**" or "**no**" depending on whether the user belongs to the **role "Administrator"**. Sort the results by username. Display all users, including the users who have no ads. Name the columns exactly like in the table below. Submit for evaluation the result grid with headers.

|  |  |  |
| --- | --- | --- |
| **UserName** | **AdsCount** | **IsAdministrator** |
| admin | 6 | Yes |
| didi | 6 | No |
| kiro | 3 | No |
| … | … | … |

6 score

### Ads by Town

Find the **count of ads for each town**. Display the **ads count** and town name or "(no town)" for the ads without a town. Display only the towns, which hold 2 or 3 ads. Sort the results by town name. Name the columns exactly like in the table below. Submit for evaluation the result grid with headers.

|  |  |
| --- | --- |
| **AdsCount** | **Town** |
| 3 | (no town) |
| 2 | Blagoevgrad |
| 3 | Dobrich |
| … | … |

5 score

### Pairs of Dates within 12 Hours -???

Consider the dates of the ads. Find among them all pairs of different dates, such that the second date is no later than **12 hours** after the first date. Sort the dates in increasing **order by the first date, then by the second date**. Name the columns exactly like in the table below. Submit for evaluation the result grid with headers.

|  |  |
| --- | --- |
| **FirstDate** | **SecondDate** |
| 2014-12-23 19:04:27.000 | 2014-12-24 05:15:37.000 |
| 2014-12-24 17:38:58.000 | 2014-12-24 20:21:11.000 |
| 2014-12-24 17:38:58.000 | 2014-12-24 23:53:17.000 |
| … | … |

5 score

## Part II – Changes in the Database

You are given a **MS SQL Server database "Ads"** holding advertisements, organized by categories and towns, available as SQL script. Your task is to **modify the database schema** and data and write SQL queries for displaying data from the database.

### Ads by Country

1. Create a **table Countries(Id, Name)**. Use auto-increment for the primary key.

Add a new column **CountryId** in the **Towns table** to link each town to some country (non-mandatory link). Create a **foreign key** between the Countries and Towns tables

1. Execute the following SQL script (it should pass without any errors):

|  |
| --- |
| INSERT INTO Countries(Name) VALUES ('Bulgaria'), ('Germany'), ('France')  UPDATE Towns SET CountryId = (SELECT Id FROM Countries WHERE Name='Bulgaria')  INSERT INTO Towns VALUES  ('Munich', (SELECT Id FROM Countries WHERE Name='Germany')),  ('Frankfurt', (SELECT Id FROM Countries WHERE Name='Germany')),  ('Berlin', (SELECT Id FROM Countries WHERE Name='Germany')),  ('Hamburg', (SELECT Id FROM Countries WHERE Name='Germany')),  ('Paris', (SELECT Id FROM Countries WHERE Name='France')),  ('Lyon', (SELECT Id FROM Countries WHERE Name='France')),  ('Nantes', (SELECT Id FROM Countries WHERE Name='France')) |

1. Write and execute a SQL command that **changes the town to "Paris" for all ads created at Friday**. ???
2. Write and execute a SQL command that **changes the town to "Hamburg" for all ads created at Thursday**.
3. Delete all ads created by user in role "**Partner**".
4. Add a **new add** holding the following information: Title="Free Book", Text="Free C# Book", Date={current date and time}, Owner="nakov", Status="Waiting Approval".!!!
5. Find the **count of ads for each town**. Display the **ads count**, the **town name** and the **country name**. Include also the ads without a town and country. Sort the results by town, then by country. Name the columns exactly like in the table below. Submit for evaluation the result grid with headers.

|  |  |  |
| --- | --- | --- |
| **Town** | **Country** | **AdsCount** |
| NULL | NULL | 3 |
| Berlin | Germany | 0 |
| Blagoevgrad | Bulgaria | 1 |
| Burgas | Bulgaria | 1 |
| … | … | … |

20 score

## Part III – Stored Procedures

You are given a **MS SQL Server database "Ads"** holding advertisements, organized by categories and towns, available as SQL script. Your task is to write some stored procedures, views and other server-side database objects and write some SQL queries for displaying data from the database.

**Important:** start with a clean copy of the "Ads" database.

### Create a View and a Stored Function

Create a **view "AllAds"** in the database that holds information about ads: **id**, **title**, **author** (username), **date**, **town** name, **category** name and **status**, sorted by id. If you execute the following SQL query:

|  |
| --- |
| SELECT \* FROM AllAds |

The result should be the table below:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Id** | **Title** | **Author** | **Date** | **Town** | **Category** | **Status** |
| 1 | Just a Joke | admin | 2014-12-24 23:53:17.000 | NULL | NULL | Published |
| 2 | 2 bed apartment to let | kiro | 2014-12-24 20:21:11.000 | Ruse | Properties | Published |
| 3 | BMW 320 for Sale | didi | 2014-12-26 12:37:50.000 | Sofia | Cars | Published |
| 4 | iPhone 4S for Sale | didi | 2014-12-27 01:56:47.000 | Dobrich | Phones | Published |
| … | … | … | … | … | … | … |

1. Using the view above, create a stored function "**fn\_ListUsersAds**" that **returns a table**, holding **all users** in descending order as first column, along with **all dates of their ads** (in ascending order) in format "yyyyMMdd", separated by "; " as second column.

If your function is correct and you execute the following SQL query:

|  |
| --- |
| SELECT \* FROM fn\_ListUsersAds() |

The result should be the table below:

|  |  |
| --- | --- |
| **UserName** | **AdDates** |
| Peter | 20141226; 20150219 |
| Nakov | NULL |
| Milena | NULL |
| Maria | 20141227; 20141231 |
| … | … |

Name the columns exactly like in the table below. Submit for evaluation the result grid with headers.

15 score

## Part V – Database Schema Design

Your task is to design a MySQL database schema, fill some data in it and write a query to retrieve some data.

### Design a Database Schema in MySQL and Write a Query

1. Design a **MySQL database "orders"** to hold products, customers and orders. **Products** hold name and price. **Customers** hold name. **Orders** hold customer, date and set of order items. **Order items** hold order, product and quantity. All tables should have auto-increment primary key – **id**.

--DROP DATABASE IF EXISTS `orders`;

CREATE DATABASE `orders` CHARACTER SET utf8 COLLATE utf8\_unicode\_ci;

USE `orders`;

DROP TABLE IF EXISTS `products`;

CREATE TABLE `products` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(45) NOT NULL,

`price` decimal(10,2) NOT NULL,

PRIMARY KEY (`id`)

);

DROP TABLE IF EXISTS `customers`;

CREATE TABLE `customers` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(45) NOT NULL,

PRIMARY KEY (`id`)

);

DROP TABLE IF EXISTS `orders`;

CREATE TABLE `orders` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`date` datetime NOT NULL,

PRIMARY KEY (`id`)

);

DROP TABLE IF EXISTS `order\_items`;

CREATE TABLE `order\_items` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`order\_id` int(11) NOT NULL,

`product\_id` int(11) NOT NULL,

`quantity` decimal(10,2) NOT NULL,

PRIMARY KEY (`id`),

KEY `fk\_order\_items\_orders\_idx` (`order\_id`),

KEY `fk\_order\_items\_products\_idx` (`product\_id`),

CONSTRAINT `fk\_order\_items\_orders` FOREIGN KEY (`order\_id`) REFERENCES `orders` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT `fk\_order\_items\_products` FOREIGN KEY (`product\_id`) REFERENCES `products` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION

);

INSERT INTO `products` VALUES (1,'beer',1.20), (2,'cheese',9.50), (3,'rakiya',12.40), (4,'salami',6.33), (5,'tomatos',2.50), (6,'cucumbers',1.35), (7,'water',0.85), (8,'apples',0.75);

INSERT INTO `customers` VALUES (1,'Peter'), (2,'Maria'), (3,'Nakov'), (4,'Vlado');

INSERT INTO `orders` VALUES (1,'2015-02-13 13:47:04'), (2,'2015-02-14 22:03:44'), (3,'2015-02-18 09:22:01'), (4,'2015-02-11 20:17:18');

INSERT INTO `order\_items` VALUES (12,4,6,2.00), (13,3,2,4.00), (14,3,5,1.50), (15,2,1,6.00), (16,2,3,1.20), (17,1,2,1.00), (18,1,3,1.00), (19,1,4,2.00), (20,1,5,1.00), (21,3,1,4.00), (22,1,1,3.00);

SELECT

p.name AS product\_name,

COUNT(oi.product\_id) AS num\_orders,

IFNULL(SUM(oi.quantity), 0) as quantity,

p.price,

IFNULL(SUM(oi.quantity) \* p.price, 0) AS total\_price

FROM

products p

LEFT JOIN order\_items oi ON p.id = oi.product\_id

LEFT JOIN orders o ON oi.order\_id = o.id

GROUP BY

p.id, p.name, p.price

ORDER BY

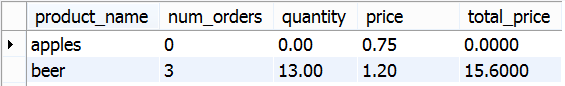
p.name

---------------------------------------------------------

1. Execute the following SQL script to load data in your tables. It should pass without any errors:

|  |
| --- |
| INSERT INTO `products` VALUES (1,'beer',1.20), (2,'cheese',9.50), (3,'rakiya',12.40), (4,'salami',6.33), (5,'tomatos',2.50), (6,'cucumbers',1.35), (7,'water',0.85), (8,'apples',0.75);  INSERT INTO `customers` VALUES (1,'Peter'), (2,'Maria'), (3,'Nakov'), (4,'Vlado');  INSERT INTO `orders` VALUES (1,'2015-02-13 13:47:04'), (2,'2015-02-14 22:03:44'), (3,'2015-02-18 09:22:01'), (4,'2015-02-11 20:17:18');  INSERT INTO `order\_items` VALUES (12,4,6,2.00), (13,3,2,4.00), (14,3,5,1.50), (15,2,1,6.00), (16,2,3,1.20), (17,1,2,1.00), (18,1,3,1.00), (19,1,4,2.00), (20,1,5,1.00), (21,3,1,4.00), (22,1,1,3.00); |

1. Write a SQL query to **list all products alphabetically** along with the **number of orders** for each product, the **quantity** sold (for all orders), the product **price**, and the **total price** (quantity \* price). Name the columns exactly like in the table below:



Submit for evaluation the result grid in CSV format, just like the sample below:

|  |
| --- |
| product\_name,num\_orders,quantity,price,total\_price  apples,0,0.00,0.75,0.0000  beer,3,13.00,1.20,15.6000  … |

20 score

DROP DATABASE IF EXISTS `orders`; --???

CREATE DATABASE `orders` CHARACTER SET utf8 COLLATE utf8\_unicode\_ci;

USE `orders`;

DROP TABLE IF EXISTS `products`;

CREATE TABLE `products` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(45) NOT NULL,

`price` decimal(10,2) NOT NULL,

PRIMARY KEY (`id`)

);

DROP TABLE IF EXISTS `customers`;

CREATE TABLE `customers` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(45) NOT NULL,

PRIMARY KEY (`id`)

);

DROP TABLE IF EXISTS `orders`;

CREATE TABLE `orders` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`date` datetime NOT NULL,

PRIMARY KEY (`id`)

);

DROP TABLE IF EXISTS `order\_items`;

CREATE TABLE `order\_items` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`order\_id` int(11) NOT NULL,

`product\_id` int(11) NOT NULL,

`quantity` decimal(10,2) NOT NULL,

PRIMARY KEY (`id`),

KEY `fk\_order\_items\_orders\_idx` (`order\_id`),

KEY `fk\_order\_items\_products\_idx` (`product\_id`),

CONSTRAINT `fk\_order\_items\_orders` FOREIGN KEY (`order\_id`) REFERENCES `orders` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT `fk\_order\_items\_products` FOREIGN KEY (`product\_id`) REFERENCES `products` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION

);

INSERT INTO `products` VALUES (1,'beer',1.20), (2,'cheese',9.50), (3,'rakiya',12.40), (4,'salami',6.33), (5,'tomatos',2.50), (6,'cucumbers',1.35), (7,'water',0.85), (8,'apples',0.75);

INSERT INTO `customers` VALUES (1,'Peter'), (2,'Maria'), (3,'Nakov'), (4,'Vlado');

INSERT INTO `orders` VALUES (1,'2015-02-13 13:47:04'), (2,'2015-02-14 22:03:44'), (3,'2015-02-18 09:22:01'), (4,'2015-02-11 20:17:18');

INSERT INTO `order\_items` VALUES (12,4,6,2.00), (13,3,2,4.00), (14,3,5,1.50), (15,2,1,6.00), (16,2,3,1.20), (17,1,2,1.00), (18,1,3,1.00), (19,1,4,2.00), (20,1,5,1.00), (21,3,1,4.00), (22,1,1,3.00);

SELECT

p.name AS product\_name,

COUNT(oi.product\_id) AS num\_orders,

IFNULL(SUM(oi.quantity), 0) as quantity,

p.price,

IFNULL(SUM(oi.quantity) \* p.price, 0) AS total\_price

FROM

products p

LEFT JOIN order\_items oi ON p.id = oi.product\_id

LEFT JOIN orders o ON oi.order\_id = o.id

GROUP BY

p.id, p.name, p.price

ORDER BY

p.name

## Exam Information

To avoid locale-specific problems, use the "**English / United States**" as your locale. The decimal point is ".", the CSV column separator is ",", the month names are in English, etc.

You are allowed to use any resources you have like Internet, software, existing code.

You are not allowed to get help from other people: Skype, ICQ, FB, email, talks, phone calls, etc. are forbidden.

Exam time: **5 hours**.