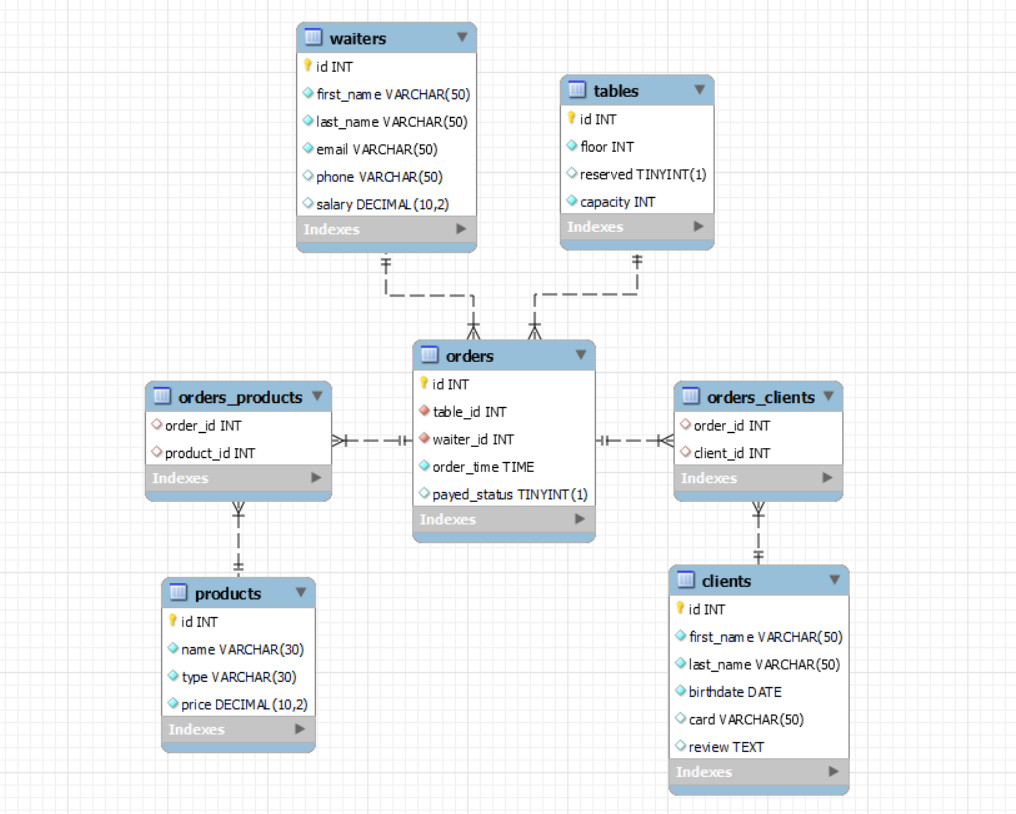
# MySQL Exam Restaurant

A very classy 3 floors restaurant has opened in the city. The owner wants to have restaurant management software that will help him and the managers run the restaurant end to end.

Help them by implementing the database structure, optimize his system and make analysis for the future management strategy of the restaurant.

## Section 0: Database Overview

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



The **restaurant\_db** **Database** needs to hold information about **orders, products, clients, waiters and tables**.

Your task is to create a database called **restaurant\_db**. Then you will have to create several **tables**.

* products – contains information about the **products**.
* clients – contains information about the **clients**.
* tables – contains information about the **tables**.
* waiters – contains information about the **waiters**.
* orders – contains information about the **orders**.
  + Each order has a table, waiter and clients.
* orders\_products – a **many** to **many** **mapping** table between the **orders** and the **products**.
* orders\_clients – a **many** to **many** **mapping** table between the **orders** and the **clients**.

## Section 1: Data Definition Language (DDL) – 40 pts

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

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

### Table Design

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

**products**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer,** from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| name | A **string** containing a maximum of **30 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**. UNIQUE** values. |
| type | A **string** containing a maximum of **30 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |
| price | **DECIMAL**, up to **10 digits**, **2** of which after the **decimal point**. | **NULL** is **NOT** permitted**.** |

**clients**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer,** from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| first\_name | A **string** containing a maximum of **50 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |
| last\_name | A **string** containing a maximum of **50 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |
| birthdate | The birth **date** of the client. | **NULL** is **NOT** permitted**.** |
| card | A **string** containing a maximum of **50 characters**. Unicode is **NOT** needed. |  |
| review | A very **long** string field |  |

**tables**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer,** from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| floor | **Integer,** from **1** to **2,147,483,647.** | **NULL** is **NOT** permitted**.** |
| reserved | Can be **true** or **false** |  |
| capacity | **Integer,** from **1** to **2,147,483,647.** | **NULL** is **NOT** permitted**.** |

**waiters**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer,** from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| first\_name | A **string** containing a maximum of **50 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |
| last\_name | A **string** containing a maximum of **50 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |
| email | A **string** containing a maximum of **50 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |
| phone | A **string** containing a maximum of **50 characters**. Unicode is **NOT** needed. |  |
| salary | **DECIMAL**, up to **10 digits**, **2** of which after the **decimal point**. |  |

**orders**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer,** from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| table\_id | **Integer,** from **1** to **2,147,483,647.** | **NULL** is **NOT** permitted**.** |
| waiter\_id | **Integer,** from **1** to **2,147,483,647.** | **NULL** is **NOT** permitted**.** |
| order\_time | **Time** that the order has been created. | **NULL** is **NOT** permitted**.** |
| payed\_status | Can be **true** or **false.** |  |

**orders\_clients**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| order\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table orders. |
| client\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table clients. |

**orders\_products**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| order\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table orders. |
| product\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table products. |

Submit your solutions in Judge on the first task. Submit **all** SQL table creation statements.

You will also be given a data.sql file. It will contain a **dataset** with random data which you will need to **store** in your **local database**. This data will be given to you so you will not have to think of data and lose essential time in the process. The data is in the form of **INSERT** statement queries.

## Section 2: Data Manipulation Language (DML) – 30 pts

Here we need to do several manipulations in the database, like changing data, adding data etc.

Select and join only tables and columns that are needed in the exercises. Any additional or less information will be considered wrong.

### Insert

You will have to **insert** records of data into the **products** table, based on the **waiters** table.

For **waiters** with **id** greater than **6**, **insert data** in the **products** table with the **following values**:

• **name** – set it to the **last name** of the waiter followed by **white space** and then **"specialty"**.

* **(last\_name + " " + "specialty")**

• **type** – set it to the **"Cocktail".**

• **price** – set it to **1%** of the **waiter salary** and round the DECIMAL always to the next largest number**.**

*(HINT: FLOOR will round the decimal to the previous whole number, but we need the opposite of FLOOR)*

### Update

Somebody made a mistake with the orders and you should correct it. Find the **orders** with **id** from **12(inclusive)** to **23(inclusive)** and **lower** their **tables id** with **1**.  
 *-* e.g*. table\_id 40 -> table\_id 39*

### Delete

**Delete** all **waiters**, who don't have any orders.

## Section 3: Querying – 50 pts

And now we need to do some data extraction. **Note** that the **example results** from **this section** use a **fresh database**. It is **highly recommended** that you **clear** the **database** that has been **manipulated** by the **previous problems** from the **DML** **section** and **insert again** the **dataset** you’ve been given, to ensure **maximum consistency** with the **examples** given in this section.

### Clients

Extract from the **restaurant\_db** system database, info about the **clients**.

**Order** the results by **birthdate** in **descending** order and **id** in **descending**;

#### Required Columns

* id (clients)
* first\_name
* last\_name
* birthdate
* card
* review

#### Example

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **first­\_name** | **last\_name** | **birthdate** | **card** | **review** |
| 88 | Eal | Scorey | 2000-10-29 | maestro | I tried to shatter it but got potato all over it. |
| 57 | Jedidiah | Thunnercliffe | 1999-09-26 | NULL | I saw one of these in Bhutan and I bought one. |
| 93 | Debi | De Souza | 1999-04-29 | visa | NULL |
| 31 | Joye | Beveridge | 1999-04-29 | maestro | I tried to pepper it but got prune all over it. |
| ... | ... | ... | ... | ... | ... |

### Birthdate

Write a query that returns: **first\_name, last\_name, birthdate** and **review** from **clients**. **Filter** clients which **don’t** have **card** and their **birthdate** is between 1978 and 1993 **inclusive**.

Show only the first **5** results and **order** them **descending** by **last\_name**, then by **id ascending**.

#### Required Columns

* first\_name
* last\_name
* birthdate
* review

#### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **first\_name** | **last\_name** | **birthdate** | **review** |
| Trudie | Swayte | 1979-10-14 | heard about this on melodic death metal radio, decided to give it a try. |
| Chantal | Sor | 1978-12-04 | My neighbor Krista has one of these. She works as a salesman and she says it looks soapy. |
| Alphard | Skelly | 1978-05-15 | My neighbor Lori has one of these. She works as a taxidermist and she says it looks whopping. |
| Marya | Niessen | 1989-06-09 | The box this comes in is 5 light-year by 6 foot and weights 17 megaton!!! |
| George | Dymocke | 1988-04-03 | i use it barely when i'm in my store. |

### Accounts

The waiters needs to have access to the new software, so they need personal accounts. Your task is to generate their usernames and passwords.

Write a query that returns: **username** and **password** for all **waiters** which are **not** fired (fired waiter is a waiter without salary). The **username** is generated by their **last name** immediately followed by their **first name** followed by the **number of characters** from **first name** and at the end "**Restaurant**". The **password** is their **email** starting from the **2nd** character to the **13th** character and then **reversed**.

**Order** by password in **descending** order.

#### Required Columns

* username (last\_name + first\_name + characters count from first\_name + "Restaurant")
* password (starting from the 2nd character to the 13th character of their email and then reversed)

#### Example

|  |  |
| --- | --- |
| **username** | **password** |
| BroadisDrusy5Restaurant | wen@3sidaorb |
| GeeringBrandon7Restaurant | ssi@9gnireeg |
| LevinChristy7Restaurant | namys@2nivel |
| … | … |
| FulgerRaffarty8Restaurant | csid@6regluf |

### Top from menu

There are many items in our menu list, but the owner wants to know which one is the best sellable item from the restaurant.

Extract from the database the **id**(product), the **name** and the **count** of **products** from all orders with this name where the **count** is greater or equal to **5**.

**Order** the results **descending** by **count** and then by **name** in **ascending**.

#### Required Columns

* id (product)
* name (product)
* count (the count of products with the same name)

#### Examples

|  |  |  |
| --- | --- | --- |
| **id** | **name** | **count** |
| 143 | Beef Minced Meat with Bulgur | 9 |
| 100 | Hot chocolate | 9 |
| 136 | Chicken Crispy Fillets 300 g. | 8 |
| … | … | … |
| 86 | Розе от Совиньон Блан Резерва | 5 |

### Availability

There are a lot of people waiting to have a dinner in the restaurant. You can help the waiters by checking the available tables in restaurant\_db.

Write a query that returns the **table\_id**, **capacity**, **count\_clients** and **availability** of all tables from the **1st** floor. **Count\_clients** is the number of people from all orders that are sitting on that table. **Availability** is based on how many people are sitting and the capacity of the table. If the capacity is **greater** than count\_clients than it should be "**Free seats**", if the capacity is **equal** to the count\_clients it should be "**Full"**, and if the capacity is **lower** than the count\_clients it should be "**Extra seats**".

**Order** the results **descending** by **table\_id**.

#### Required Columns

* table\_id
* capacity
* count\_clients (is the number of people from all orders that are sitting on that table)
* availability (based on how many people are sitting and the capacity of the table)

#### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **table\_id** | **capacity** | **count\_clients** | **availability** |
| 29 | 5 | 5 | Full |
| 18 | 6 | 6 | Full |
| 13 | 9 | 6 | Free seats |
| … | … | … | … |
| 2 | 10 | 19 | Extra seats |

## Section 4: Programmability – 30 pts

The time has come for you to prove that you can be a little more dynamic on the database. So, you will have to write several procedures.

### Extract bill

Create a **user defined function** with the name **udf\_client\_bill(full\_name VARCHAR(50))** that receives a **customer’s full name** and returns the total price of products he ordered;

#### Required Columns

* first\_name (client)
* last\_name (client)
* bill (udf\_client\_bill) (should be DECIMAL(19,2))

#### Example

|  |  |  |
| --- | --- | --- |
| **Query** | | |
| SELECT c.first\_name,c.last\_name, udf\_client\_bill('Silvio Blyth') as 'bill' FROM clients c  WHERE c.first\_name = 'Silvio' AND c.last\_name= 'Blyth'; | | |
| first\_name | last\_name | bill |
| Silvio | Blyth | 96.90 |

### Happy hour

Create a stored procedure udp\_happy\_hour which accepts the following parameters:

* type (VARCHAR(50))

Extracts data about the **products** from the given **type** and reduces the **prices** by **20%** of all **products** which have price higher or equal to **10.00** and are from the given **type**.

#### Result

|  |
| --- |
| **Query** |
| CALL udp\_happy\_hour ('Cognac'); |
| This execution will update 1 product – Martell VS Single Distillery F |
| Result |
| Brandy Ararat 5YO Armenia|6.00 -> 6.00  Brandy Sarajishvili VS, Georgi|8.00 -> 8.00  Martell VS Single Distillery F|10.00 -> 8.00 |