MySQL Exam Preparation – 5 February 2024

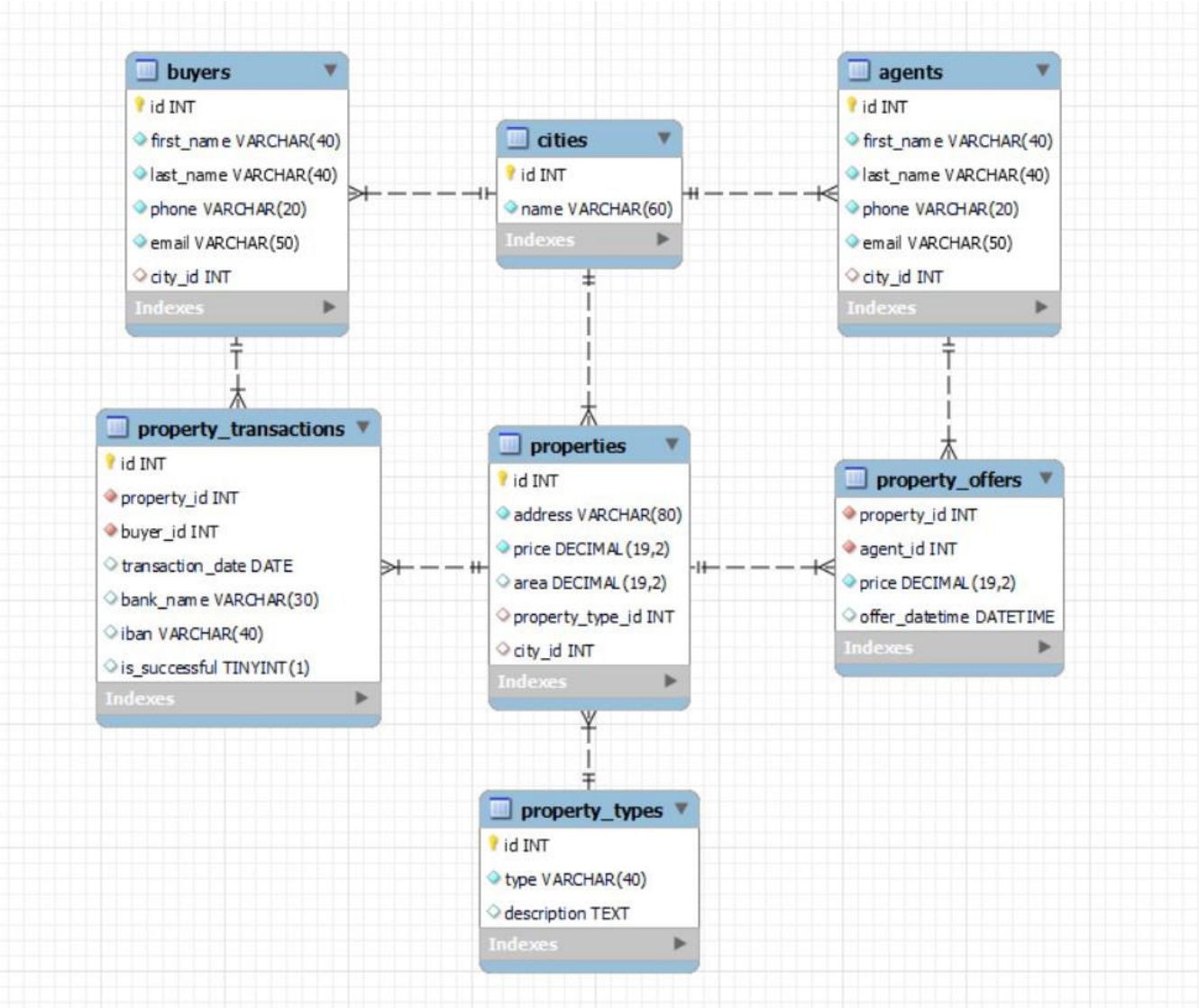
Real Estate DB

Link към Database Basics MySQL

A real estate database is an indispensable tool for managing and optimizing real estate operations. With this task, the objective was to create a comprehensive database that encompasses various aspects of the real estate industry. The database includes tables for cities, property types, properties, agents, buyers, property offers, and property transactions. This centralized system enables efficient storage and retrieval of crucial information related to cities, property details, agents, buyers, and the offers and transactions associated with properties. By implementing this database, real estate professionals can effectively manage property listings, track transactions, and ensure seamless communication between agents and buyers. This powerful tool enhances productivity, streamlines processes, and fosters a safe and efficient real estate environment.

Section 0: Database Overview

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



The **real_estate_db's Database** needs to hold information about **agents**, **buyers**, **cities, properties**, **property_types, property_transactions** and **property_offers**.

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

- **cities** – contains information about the **cities**.
- **property_types** – contains information about the different **types** of **property**.
- **properties** – contains information about the **properties**.
    - Each **property** has a **city** and **property_type**.
- **agents** – contains information about the **agents**.
    - Each **agent** has a **city**.
- **buyers** – contains information about the **buyers**.
    - Each **buyer** has a **city**.
- **property_offers** – a **many** to **many mapping** table between the **properties** and the **agents** that contains information about the **properties offers**.
    - Each **property_offer** has a **property** and **agent**.
- **property_transactions**– contains information about each **property transaction**.
    - Each **property_transaction** has a **property** and **buyer**.

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

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

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

01. Table Design

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

cities

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 **60 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted**.** **UNIQUE** values.

property_types

Column Name	Data Type	Constraints
`id`	**Integer,** from **1** to **2,147,483,647.**	**Primary Key** **AUTO_INCREMENT**
`type`	A **string** containing a maximum of 4**0 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted**.** **UNIQUE** values.
`description`	A very **long** string field	

properties

Column Name	Data Type	Constraints
`id`	**Integer,** from **1** to **2,147,483,647.**	**Primary Key** **AUTO_INCREMENT**
`address`	A **string** containing a maximum of **80 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted. **UNIQUE** values.
`price`	**DECIMAL**, up to **19 digits**, **2** of which are after the **decimal point**.	**NULL** is **NOT** permitted.
`area`	**DECIMAL(19, 2)**	
`property_type_id`	**Integer,** from **1** to **2,147,483,647.**	Relationship with table **`property_types`**.
`city_id`	**Integer,** from **1** to **2,147,483,647.**	Relationship with table **`cities`**.

agents

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 **40 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted.
`last_name`	A **string** containing a maximum of **40 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted.
`phone`	A **string** containing a maximum of **20 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted. **UNIQUE** values.
`email`	A **string** containing a maximum of **50 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted. **UNIQUE** values.
`city_id`	**Integer,** from **1** to **2,147,483,647.**	Relationship with table **`cities`**.

buyers

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 **40 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted.

`last_name`	A **string** containing a maximum of **40 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted**.**
`phone`	A **string** containing a maximum of **20 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted**.** **UNIQUE** values.
`email`	A **string** containing a maximum of **50 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted**.** **UNIQUE** values.
`city_id`	**Integer,** from **1** to **2,147,483,647.**	Relationship with table `cities`.

property_offers

Column Name	Data Type	Constraints
`property_id`	**Integer,** from **1** to **2,147,483,647.**	Relationship with table `properties`. **NULL** is **NOT** permitted.
`agent_id`	**Integer,** from **1** to **2,147,483,647.**	Relationship with table `agents`. **NULL** is **NOT** permitted.
`price`	**DECIMAL**, up to **19 digits**, **2** of which after the **decimal point**.	**NULL** is **NOT** permitted**.**
`offer_datetime`	The **date** and **time** of the offers creation.	

property_transactions

Column Name	Data Type	Constraints
`id`	**Integer,** from **1** to **2,147,483,647.**	**Primary Key** **AUTO_INCREMENT**
`property_id`	**Integer,** from **1** to **2,147,483,647.**	Relationship with table `properties.` **NULL** is **NOT** permitted.
`buyer_id`	**Integer,** from **1** to **2,147,483,647.**	Relationship with table `buyers`. **NULL** is **NOT** permitted**.**
`transaction_date`	The **date** of the transaction.	
`bank_name`	A **string** containing a maximum of **30 characters**. Unicode is **NOT** needed.	

`iban`	A **string** containing a maximum of **40 characters**. Unicode is **NOT** needed.	**UNIQUE** values.
`is_successful`	Can be **true** or **false**	

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.

02. Insert

You will have to **insert** records of data into the **property_transactions** table.

The new data will be based on **property_offers** with **agent_id** equal or less than **2**. **Insert data** in the **property_transactions** table with the **following values**:

- **property_id** – set it to the **agent_id** plus the **days** of the **offer's datetime**.
- **buyer_id** – set it to the **agent_id** plus the **month** number of the **offer's datetime**.
- **transaction_date**– set it to the **date** only of the **offer's datetime**.
- **bank_name** – set it to **"Bank"** followed by whitespace and then followed by **agent_id**.
- **iban** – set it to **"BG"** followed by **price** and then followed by **agent_id**.
- **is_successful** – set it to **true**.

03. Update

There are some tax frauds and you have to correct the price for some properties. You must **reduce** the **price** by **50 000** for all **properties** that cost more or equal to **800 000**.

04. Delete

Delete all **property_transactions** that are **not successful**.

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.

05. Agents

Extract from the **real_estate_db**, info about the **agents**.

Order the results by **city_id** in **descending** and then by **phone** in **descending**.

Required Columns

- **id**
- **first_name**
- **last_name**
- **phone**
- **email**
- **city_id**

Example

id	first_name	last_name	phone	email	city_id
19	Martin	Penchev	679-129-3977	mpenchev@hostgator.com	8
11	Maud	Mulvany	830-721-8209	mmulvanya@cbsnews.com	7
20	Wesley	Grishaev	337-589-8538	wgrishaevj@timesonline.co.uk	7
...	...				

06. Offers from 2021

Write a query that returns: **property_id**, **agent_id**, **price** and **offer_datetime** from table **property_offers**. **Filter** offers only from 2021 **year**.

Order the results **ascending** by **price** and show only the first **10** results.

Required Columns

- **property_id**
- **agent_id**
- **price**
- **offer_datetime**

Example

property_id	agent_id	price	offer_datetime
24	2	46056.22	2021-10-07 21:23:16
8	17	120667.24	2021-05-18 19:46:17
37	11	138723.19	2021-09-29 09:01:40
…	…	…	…

07. Properties without offers

Some properties are not included in offers and don't have an agent.

Write a query that returns: **agent_name** and **price** for all properties that are not included in any **offer**.

To find their **agent_name** you must take the first **6** letters from the **address** and to find the offered **price** you need to get the **number of characters** in the **address** and multiply it by 5430.

Order by **agent_name** in **descending** order and then by **price** in **descending** order.

Required Columns

- `agent_name (first 6 characters from address)`
- `price (number of characters in the address multiplied by 5430`

Example

agent_name	price
Singel	54300
Rue de	86880
Rue de	76020
Hyden	97740
…	…
Alexan	124890

08. Best Banks

Our popular real estate app is set to highlight the top banks that have facilitated 9 or more transactions within our database.

Extract from the database, the **banks** that have **9** or **more ibans** used for **transactions**.

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

Required Columns

- **bank_name**
- **count**

Examples

bank_name	count
Central Savings	11
Global Trust Bank	9
United Financial	9

09. Size of the area

From the database extract the **address** and **area** and assign the **size**. If it is less or equal 100 is "**small**", "**medium**" is lesser or equal to **200**, "**large**"is lesser or equal to **500** and above **500** is "**extra large".**

Order the results **ascending** by **area** and then by **address** in **descending**.

Required Columns

- **address**
- **area**
- **size (small is lesser or equal 100, medium is lesser or equal to 200, large is lesser or equal to 500 and above 500 is extra large)**

Example

address	area	size
Naschmarkt 5	60.00	small
Praterstrasse 6	70.00	small
12 Rue de Rivoli	75.5	small
. . .	. . .	. . .
10 St. James's Square	2500.00	extra large

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.

When submitting your code in Judge paste only the CREATE code and be sure it is without DELIMITER change.

10. Offers count in a city

Create a **user defined function** with the name **udf_offers_from_city_name (cityName VARCHAR(50))** that receives a **city name** and returns the total number of **offers** from that city.

Required Columns

- **offers_count (udf_offers_from_city_name)**

Example

Query
`SELECT udf_offers_from_city_name('Vienna) AS 'offers_count';`
offers_count
10

Query
`SELECT udf_offers_from_city_name('Amsterdam`) AS 'offers_count';`
offers_count
9

11. Special Offer

The real estate agents want to make special offers for their loyal clients. Your task is to find all **offers** from an **agent** and **reduce** the **prices** by **10%**.

Create a stored procedure **udp_special_offer** which accepts the following parameters:

- **first_name VARCHAR(50)**

Result

Query
`CALL udp_special_offer('Hans');`
This execution will update all offers prices from agent with first name '**Hans**'

Result

first_name	price before	->	price after
Hans	360772.54	->	**324695.29**
Hans	609566.40	->	**548609.76**
Hans	439936.52	->	**395942.87**
Hans	325774.30	->	**293196.87**
Hans	707559.30	->	**636803.37**