CPSC 2221 - Database Systems

Group Project - Implementation of a Relational Database

Project Title:	Real Estate Database
Project Milestone:	4(Complete Project Files)

#	Student Name	Student ID	Email Address
1	Loreena Alfonso	100325280	lalfonso00@mylangara.ca
2	Sonia Arora	100323363	sarora29@mylangara.ca
3	Harneet Kaur	100333069	h006@mylangara.ca

By keying our names and student IDs in the above table, we certify that the work submitted with this cover page was performed solely by those whose names and student IDs are included above.

Also, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Langara College.

INSTRUCTIONS

1. In our Database connections we have username and password set to “root” as in the PHP tutorial. See below:
   $servername = "localhost";
   $username = "root";
   $password = "root";
   $database = "realestate";
2. Make a new database called ‘realestate’ on phpmyAdmin.
3. In our zip, there’s a file called “CREATE_INSERT_Script_realestate.sql”. This file must be imported into the ‘realestate’ database.
4. Please use chrome otherwise some images may be blocked by virus control on your browser.
5. Once imported, go to the link localhost/Group14/index.php to view our main page
6. In our zip, there’s also a file called “DROP_Script_realestate.sql”. This file can be used to run all the commands to drop our tables and / or the database.

PROJECT ACCOMPLISHMENTS

Our project accomplished a simplified mock-Zillow application where a user can register as a buyer or seller, list a house and search for houses. Users can search the house by price, by city, by number of beds and baths or by any combination of these. In addition, an agent can update their office information, provided they know their ID. A seller can also delete their listing if they choose. There is an admin view where the administrator can see which sellers have not made a listing yet, in case he or she wants to send an invite to them to post a listing.

Looking back, we had ambitious plans and it is apparent in the website that we built. We expected to have more information used out of our seller and buyer tables, including an exchange. However, despite having to simplify our project from our initial plan, we still believe that we accomplished the essence of what we set out to do. We have implemented all of the queries required and have a functional database system that displays houses for sale in the Greater Vancouver Area.

LIST OF SQL QUERIES USED

1. *Projection query :*

   This projection query selects all of the distinct cities found in our agent table, we then use it to run a select statement to find our ‘top’ agent in each distinct city.

   SELECT DISTINCT City FROM agent_r3

2. *Selection query :*

   This selection query returns all the rows from the completelisting view.

   *SELECT * FROM completelisting;*

3. *Join query :*

   Join Query happens in view Creations

   This JOIN allows for the display of content from our two agent tables to execute the projection query above and display useful data about each agent.

   CREATE VIEW agentview2 As Select agent_r3.Agent_ID, agent_R3.email, agent_r3.Phone_no, agent_r3.Office_loc, agent_r3.City, agent_r2.A_img, agent_r2.A_Name FROM agent_r3 INNER JOIN agent_r2 On agent_r3.Agent_ID = agent_r2.Agent_ID

   This JOIN allows for the display of content from 4 separate tables to be accessed when we need all of the information about a house.

   CREATE VIEW completeListing2 AS SELECT h1.Postal_code,

   h1.City,

   h2.House_no,

   h2.Address,

   h1.Price_per_sqft,

   h2.H_type,

   h2.H_size,

   p.Price,

   p.Thumbnail,

   p.List_ID,

   f2.Beds,

   f2.Baths FROM

   (((

   (house_r1 h1 INNER JOIN house_r2 h2 ON h1.Price_per_sqft= h2.Price_per_sqft)

INNER JOIN postedlisting p ON p.Price_per_sqft = h2.Price_per_sqft)

INNER JOIN feature_r2 f2 ON p.House_no = f2.House_no AND p.Address = f2.Address AND h1.City = f2.City)

INNER JOIN feature_r1 f1 ON f1.City_groupID = f2.City_groupID);

4. *Division query :*

   This query gives the seller id's of all the sellers such that there is no house that is posted by those sellers.

   SELECT s.User_ID FROM seller s WHERE NOT EXISTS (SELECT * from house_r2 h2 WHERE EXISTS (SELECT p.User_ID FROM postedListing p WHERE s.User_ID = p.User_ID AND h2.House_no = p.House_no));

5. *Aggregation query :*

   This query executes a count() on all of the results that match the user's dynamic searches. The variables represent the pieces of the query that may change depending on what options the user has selected. This is helpful as it displays the number listings that meet the search criteria.

   SELECT COUNT(*) FROM completelisting $wherePriceSQL $whereBedSQL $whereBathsSQL $whereCitySQL;

6. *Nested aggregation with group-by*

   This query calculates the current average price of a listing in each city and we are displaying it in a buyer's guide above the search results

   SELECT AVG(Price), City FROM completelisting

   GROUP BY City

   ORDER BY AVG(City) DESC

7. *Delete operation :*

   This query deletes the seller data and his posted listing using an on delete cascade constraint. It is helpful if the user doesn't want his house listing to be posted on our website anymore.

   DELETE FROM seller WHERE User_ID ='$UserID'

8. *Update Operation*

   This query allows the agent to update their information provided that the agent already knows his/her agent id.

   UPDATE Agent_R2 SET Email='$Email', Phone_no='$Phone', Office_loc='$Office' WHERE Agent_ID='$ID'