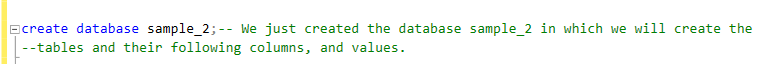
Project 3.

1-Insert at least three tables into the database sample\_2. Create at least three columns per table, be creative when it comes to the data types of your variables. In addition, insert 10 records per table.

2- Define a primary key for each table and make use of the foreign concept in a query.

First, we created the sample\_2 database in which we will create the four tables, followed by their columns and respective values.

By using create database sample\_2 query we just created the database sample\_2 in which we will create the tables and their following columns, and values.



Then we connect to the sample\_2 database that we just created by using use sample\_2;





In this case we are going to create tables for a real estate company with the following tables: **Agents,**

**Customers, Properties, and Transactions\***

First, let’s start with the **Agent Table.**

Graphical user interface, text, application

Description automatically generated

As we can see we created the table Agents with the following columns: Emp\_id as our primary key with data type char, Agent\_fname with data type char(25), Agent\_lname data type (25), and contact\_info with data type char(14).

**Customers Table.**

Graphical user interface, text, application

Description automatically generated

As we can see we successfully created the customer table with the following columns: Customer\_id as our primary key with data type(20), customer\_fname with data type char(25) , customer\_lname with data type char(25), zip\_code with int data type , contact\_no, and emp\_id as our foreign key referencing the emp\_id of our Agents table.

**Properties Table.**

Graphical user interface, text, application

Description automatically generated

As we can see we successfully created the properties table with the following columns: Property\_ID as our primary key, Property\_description, Property\_city, Property\_address, zip\_code , Property\_bedrooms, property\_bathrooms, size\_sqft, property\_enter\_date, listed\_price, and emp\_id. In addition, we added a constraint due to the fact that each property belong to only one agent, which in this case is identified by the emp\_id.

**Transactions.**

**Text

Description automatically generated**

As we can see the Transactions Table was created successfully and has the following columns:

Customer\_id, Property\_id, emp\_id, sales\_price, and enter\_date. In addition, we added the following constraint to make references of the customers, the properties, and the agent involved in the transaction.

Now we are going to insert the records into each table.

First, let’s start by inserting values into our agent’s table:

Text

Description automatically generated

As we can see we successfully added the values into the Agents table.

Now let’s add the respective records for our Customer Table

Text

Description automatically generated

Now let’s add records to our properties table.

Text

Description automatically generated with low confidence

We use the select \* from properties; to verify we added the records successfully

A screenshot of a computer

Description automatically generated with medium confidence

Now we add the values into our transactions table

Text

Description automatically generated with medium confidence

Table

Description automatically generated

Now by using the select \* from transactions we verify we added the values successfully.

3-Run at least three queries combining the Select statement and the where clause to obtain responses from our database sample\_2.

By using the query:

* Select \* from properties where listed\_price between 3000000 and 4000000;

Graphical user interface, application

Description automatically generated

We get the all the records of the properties in which the listed price is between 3000000 and 4000000.

Which are: 18975 Collins Ave with a listed price of 3450000, 1777 Datonia Rd with a listed price of 335000, and 17121 Collins Ave with a listed price of 310000.

* SELECT transactions.transaction\_id, transactions.customer\_id, transactions.sales\_price FROM transactions JOIN properties

ON transactions.property\_id=properties.property\_id

WHERE listed\_price=sales\_price;

By using this query we are asking the database to gives us the transaction id, the customer\_id , and the sales price from the transaction table joining the properties table in which the listed price equals the sales price.

Graphical user interface, text, application

Description automatically generated

We came to the conclusion that these customers were the only ones who purchased the properties in the database at the original listing price.

* Select properties.property\_description,properties.property\_address,properties.listed\_price, properties.property\_enter\_date

From properties Join transactions

On properties.property\_id=transactions.property\_id

Where listed\_price=sales\_price+(listed\_price\*0.10);

By using this query we are asking the database to select the properties description,

address, listed price, and property enter date joining the transactions table where the listed price is 10% above the sales price.

Graphical user interface, text, application

Description automatically generated

We came to the conclusion that these properties were the only ones that were sold at 10% less of the listing price.

4-Define a new column within a table that is the result of an operation using a column that already existed in the table. (hint: the budget example)

A picture containing logo

Description automatically generated

We create a new column named old price into the Transactions tables

Text

Description automatically generated

We use the select \* from transactions to confirm the old price column has been added into the transactions table successfully

Table

Description automatically generated

5-Create a temporary table resulting from retrieving two columns from an existing table.

By using the query below, we create a temporary table selecting the columns property\_address , and listed price from the properties table.

* select property\_address, listed\_price

into #propertytemp

from properties;

Graphical user interface, application

Description automatically generated

By using the select \* from #propertytemp we verify that the columns property\_address, and listed\_price from the properties table were successfully added into the #propertytemp table.

6-Use a bcp command to export data from one of your tables in the project database to an output file of your convenience. In this case, I am using the sample\_2 database and the table that I would like to export is Properties.

Type this into the command prompt

bcp sample\_2.dbo.Properties out Properties.txt -T –c

Text

Description automatically generated

First, We go to C:\Users\HP to check the table Properties was copied successfully

Graphical user interface, text

Description automatically generated with medium confidence

Text

Description automatically generated with medium confidence

As we can see the column Properties was copied successfully into a text file.

7- Use the sqlcmd utility to query your database from the command prompt (hint: example 15.5)

First, let us start the command prompt and type:

sqlcmd

Use Sample\_2;

Select \* from Agents;

: Exit (select @@rowcount)

Text

Description automatically generated

8-Backup the sample\_2 database and compresses the backup file.

Let us now backup the database using the object explorer and then lets us compress the backup file. I will compress the database named Sample\_2.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

To verify the database was successfully saved we go to the following address

Graphical user interface, table

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, table

Description automatically generated

Graphical user interface, text, application

Description automatically generated

As we can see the backup of sample\_2 was a success. Now, we are going to compress it into a zip file.

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated