

# ETL PROJECT REPORT – GROUP 3

Impact of tourism and economy on the “PUEBLOS MÁGICOS”  
program



19/10/2020

Adriana Avalos Vargas, Edwin Rosales & Jimi Jenneskens

## Table of Contents

Introduction .....	2
1. Extract .....	3
1.1 Magic Towns .....	3
1.2 The gross domestic product.....	3
1.3 Weather .....	4
2. Transform.....	5
2.1 Magic Towns table and State table.....	5
2.2 The gross domestic product table .....	6
2.3 Weather table .....	7
3. Load.....	9
3.1 Relationship diagram .....	9
3.2 Load data into database.....	9
4. Queries.....	11
4.1 Average rainfall per Magic Town .....	11
4.2 Average State GDP per Magic Town .....	12
4.3 Magic Towns per state .....	13

## Introduction

The “*Pueblos Mágicos*” program has the objective to transform Mexico, around the 2030, in a leading country in tourism activity. To achieve this, the government proposed as a strategy to recognize tourism as a key element in Mexico's economic development. Diversify tourism products and develop new markets. Promote tourism companies to be competitive nationally and internationally. Develop tourism while respecting natural, cultural and social environments, in this context the program “*Pueblos Mágicos*” was created.

According to the **SECTUR**, a magical town (“pueblo mágico”) is a locality that has symbolic attributes, legends, history, transcendent facts, everyday life, etc. In short, they possess magic that appears in each one of their socio cultural manifestations, and today they can be perceived as a great opportunity for tourist use. The Magical Towns Program contributes to revalue a group of populations in the country that have always been in the collective imagination of the nation as a whole and that represent fresh and different alternatives for national and foreign visitors.

### Proposal of an ETL project

Since the main objective of this program is to convert Mexico in leading country on tourism, then it is necessary to create a unique database that captures how does this program has impacted the different localities that has access to this program.

To do so several data sources has to be consulted and aggregated.

The actual list of localities considered as Magic town.

Its contribution to the gross domestic product associated with the Tourism activity at the state and national level since the creation of the first magic town (this info is available yearly).

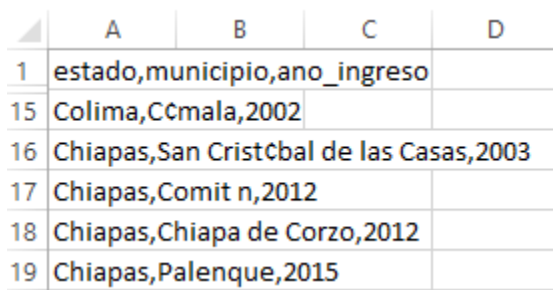
Also, it could be important to characterize the weather in each Magic town, in order to do a further analysis about the impact of this variable on the success of the program en each Magic Town.

## 1. Extract

Three different sources have been used to compile the database. The first source is a dataset with the Magic Towns and the date it has entered as a magic town. The second source is a dataset with the yearly gross domestic product per state based on the influence of tourism, which is useful to determine the influence on the gross domestic product before and after entering as a magic town. The last source is a dataset with the average weather per year per state, which can be used to investigate if the weather influences the popularity of the Magic Town. The sources are explained in the following chapters.

### 1.1 Magic Towns

The first source is a CSV file, which contains three columns with “State”, “Town” and “Year entered” per magic town. An example of the CSV file can be found in figure 1.



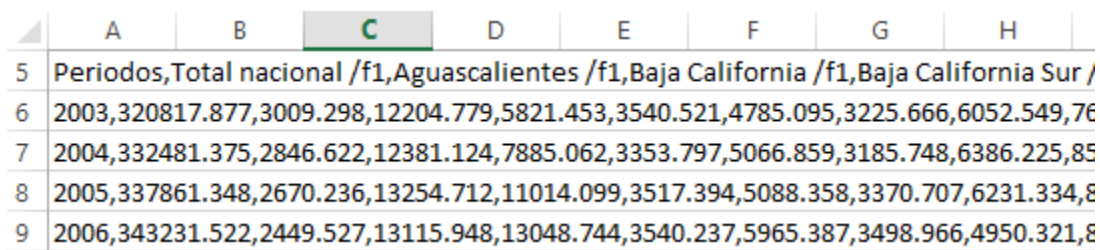
	A	B	C	D
1	estado,municipio,ano_ingreso			
15	Colima,Ccmala,2002			
16	Chiapas,San Cristóbal de las Casas,2003			
17	Chiapas,Comit n,2012			
18	Chiapas,Chiapa de Corzo,2012			
19	Chiapas,Palenque,2015			

Figure 1: Screenshot magic towns CSV

The source can be found here: <https://datos.gob.mx/busca/dataset/localidades-que-cuentan-con-el-nombramiento-de-pueblo-magico-dggd>.

### 1.2 The gross domestic product

The second source is a CSV file with the yearly gross domestic product based on tourism from 2003 till 2018, which contains the “State”, “Gross domestic product” and the “Year”. This information is only available on state level, so unfortunately it cannot be used on Magic Town level. An example of the CSV file can be found in figure 2.



	A	B	C	D	E	F	G	H
5	Periodos,	Total nacional /f1,	Aguascalientes /f1,	Baja California /f1,	Baja California Sur /f1,	Baja California Sur /f1,	Baja California Sur /f1,	Baja California Sur /f1,
6	2003,	320817.877,	3009.298,	12204.779,	5821.453,	3540.521,	4785.095,	3225.666,
7	2004,	332481.375,	2846.622,	12381.124,	7885.062,	3353.797,	5066.859,	3185.748,
8	2005,	337861.348,	2670.236,	13254.712,	11014.099,	3517.394,	5088.358,	3370.707,
9	2006,	343231.522,	2449.527,	13115.948,	13048.744,	3540.237,	5965.387,	3498.966,

Figure 2: Screenshot gross domestic product CSV

The source can be found here: <https://www.inegi.org.mx/sistemas/bie/>

### 1.3 Weather

The third source are twenty separated excel files per topic with tables of the yearly average rain, minimum temperature, maximum temperature and the average temperature per state. This information is only available on state level, so unfortunately it cannot be used on Magic Town level. An example of an excel file can be found in figure 3.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	ENTIDAD	ENE	FEB	MAR	ABR	MAY	JUN	JUL	AGO	SEP	OCT	NOV	DIC	ANUAL
2	AGUASCALIENTES	23,3	25,0	27,4	29,8	30,6	28,5	28,7	28,0	27,8	26,7	25,0	22,3	26,9
3	BAJA CALIFORNIA	22,6	21,7	22,3	27,1	29,2	31,8	32,6	33,8	32,4	25,7	21,3	22,8	26,9
4	BAJA CALIFORNIA SUR	27,0	27,5	27,2	30,4	31,4	33,9	35,6	35,8	34,8	30,4	26,4	26,8	30,6
5	CAMPECHE	29,1	30,9	34,3	36,4	36,5	33,9	34,6	32,9	32,9	31,2	31,1	28,3	32,7
6	COAHUILA	22,4	25,6	29,0	31,1	34,3	32,5	34,5	32,7	31,9	26,6	22,6	19,6	28,6
7	COLIMA	32,3	32,6	32,2	33,8	33,5	33,4	33,2	33,6	32,1	32,8	33,0	32,0	32,9

Figure 3: Screenshot weather PDF

The sources can be found here: <https://datos.gob.mx/busca/dataset/precipitacion>

<https://datos.gob.mx/busca/dataset/temperatura-promedio-excel>

<https://datos.gob.mx/busca/dataset/temperatura-minima-excel>

<https://datos.gob.mx/busca/dataset/temperatura-maxima-excel>

## 2. Transform

To be able use the data for our database the data has to be cleaned and transformed, so that it can be used correctly for the purpose of the database. The three sources have been transformed into correctly formatted tables and an extra table named 'States' has been created to be able to make a relationship between all the tables. This part is explained per table in the following chapters.

### 2.1 Magic Towns table and State table

As shown in figure 1, the original CSV file contains spelling mistakes in the states and town, because the CSV file cannot contain special characters. Therefore, first the names of the states and towns have been corrected manually to without special characters.

Another problem is that the states are spelled differently throughout the sources and cannot have a relationship in the database via this way. Therefore, an extra table has been created named 'States\_ids', to be able to create a relationship between the different sources/tables. All the unique states have been stored in a dataframe and via a 'for loop' unique ids have been added. In figure 4 is an example of this table shown. This table has been stored as a CSV file.

```
#Lets create a numeric id
seq_s = [i+1 for i in range(32)]
#Lets add it to the df
states['id_state'] = seq_s
states = states[['id_state', 'state', 'abbr_state']]
states.rename(columns={"id_state": "id"}, inplace=True)
#call the df
states
```

	id	state	abbr_state
0	1	AGUASCALIENTES	AGS
1	2	BAJA CALIFORNIA	BC
2	3	BAJA CALIFORNIA SUR	BCS
3	4	CAMPECHE	CAMP
5	5	CHIAPAS	CHIS
4	6	CHIHUAHUA	CHIH
8	7	CIUDAD DE MEXICO	DF

Figure 4: States table

```
# drop state column
dataset_df = dataset_df.drop(['state_mt'], axis = 1)

dataset_df
```

	id	town	year_entered	id_state
0	0	Real de Asientos	2006	1
1	1	Calvillo	2012	1
2	2	San Jose de Gracia	2015	1
3	3	Tecate	2012	2
4	4	Todos Santos	2006	3

Figure 5: Magic towns table

The id\_state has been added to the Magic Towns manually. The headers in the original CSV file, as shown in figure 1, are in Spanish and therefore, have been changed to English headers. The states column has been dropped, as this information can be retrieved via the States table. Also the index has been reset and the new column has been named id, so it creates unique ids per town. In figure 5 is an example of the Magic Towns table shown. This table has been stored as a CSV file.

## 2.2 The gross domestic product table

The dataframe of the original CSV is not ready to be used yet. This one has to be modified in order to be able to use it as a table for a database. In figure 6 is an example shown of the dataframe of the original dataframe.

	ano	totalnacional	aguascalientes	baja california	baja california sur	campeche	coahuila	colima
0	2003	320817.877	3009.298	12204.779	5821.453	3540.521	4785.095	3225.666
1	2004	332481.375	2846.622	12381.124	7885.062	3353.797	5066.859	3185.748
2	2005	337861.348	2670.236	13254.712	11014.099	3517.394	5088.358	3370.707
3	2006	343231.522	2449.527	13115.948	13048.744	3540.237	5965.387	3498.966

Figure 6: Dataframe of the original gross domestic product CSV

First index has been set to 'ano' and then the axes of the index and the columns have been swapped, so that the states are on the rows and the index and the years in the column. Then the states have been sorted on values, so it has the same order as the State\_ids table and an extra column named 'id\_state' has been added with counting from id 1 to 32, so that it can have a relationship with the other tables in the database. In figure 7 is a visible of the new dataframe visible.

	id	state	2003	2004	2005	2006	2007	2013	2014	2015	2016	2017	2018	id_state
1	1	aguascalientes	3009.298	2846.622	2670.236	2449.527	2376.980	2902.186	3013.849	3390.240	3669.932	3928.135	4396.160	1
2	2	baja california	12204.779	12381.124	13254.712	13115.948	13091.387	10893.632	11263.834	12024.583	12641.127	12643.593	11137.582	2
3	3	baja california sur	5821.453	7885.062	11014.099	13048.744	15306.901	15543.510	14647.114	17284.922	18672.983	20919.630	23710.001	3
4	4	campeche	3540.521	3353.797	3517.394	3540.237	3436.569	4254.560	4821.917	4434.030	4174.454	3920.209	4106.851	4
7	7	chiapas	6052.549	6386.225	6231.334	4950.321	6006.764	5361.347	5228.832	6077.716	5992.626	5977.213	5928.178	5
8	8	chihuahua	7612.652	8531.909	8293.606	8912.666	7609.848	7298.381	6506.445	7980.328	7914.189	7702.155	7850.709	6

Figure 7: Dataframe with changed axes and id\_state column

Then the values in the column state have been changed to capital letters only, as in the other tables. A list of year 2003 till 2018 has been created, in order to create a new dataframe with a 'for loop'. A 'for loop' has been created to create a new dataframe based on 'id\_state', 'states', 'tourism\_gdp' and 'year'. The states column has been dropped, as this info can be found via the States table.

The 'for loop' can be found in figure 8. The new and final gross domestic product table can be found in figure 9 on the next page. This table has been stored as a CSV file.

```
#Lets create an empty df
gdp_df = pd.DataFrame(columns = ['id_state', 'states', 'tourism_gdp', 'year'])
gdp_df
```

```
id_state  states  tourism_gdp  year
```

```
#Loop to rewrite the df
for year in years:
    syear = f"{year}"
    #Select the columns of interest as a list
    data = [new_df["id_state"], new_df["state"], new_df[syear]]
    #Create the name of the columns
    headers = ['id_state', 'states', 'tourism_gdp',]
    #Coerce in a df
    aux = pd.concat(data, axis=1, keys=headers)
    #Add a column with the year
    aux["year"] = [year] * 32
    #Concatenate in the empty df
    gdp_df = pd.concat([gdp_df, aux])
```

Figure 8: The 'for loop' to create the new dataframe

	id_state	tourism_gdp	year
1	1	3009.298	2003
2	2	12204.779	2003
3	3	5821.453	2003
4	4	3540.521	2003
7	5	6052.549	2003
...	...	...	...
28	28	7325.560	2018
29	29	1427.076	2018
30	30	11470.447	2018
31	31	5759.519	2018
32	32	2747.637	2018

512 rows × 3 columns

Figure 9: The gross domestic product table

### 2.3 Weather table

For the topics yearly rain, minimum temperature, maximum temperature and average temperature there were 20 separated yearly excel files per topic. These files have been concatenated into a CSV file per topic via a 'for loop' and the id\_state column has been added. An example of this loop can be found in figure 10 on the next page. An example of the result, which has been stored as a CSV file, can be found in figure 11 on the next page.



```
#Create a loop to read the excel file for each year, tra
#Generate a loop to read the file and combine it in a u
for year in years:
    #Create the path to each excel file
    path =f"{year}Tmed.xlsx"
    #Read the excel file
    prueba = pd.read_excel (path)
    #Drop national observation
    prueba.drop(prueba.tail(1).index, inplace = True)
    #Select the columns of interest as a list
    data = [prueba["ENTIDAD"], prueba["ANUAL"]]
    #Create the name of the columns
    headers = ["state", "average_temperature"]
    #Coerce in a df
    aux = pd.concat(data, axis=1, keys=headers)
    #Add a column with the year
    aux["year"] = [year] * 32
    aux.sort_values(by=['state'], inplace=True)
    #Lets create a numeric id
    seq_s = [i+1 for i in range(32)]
    #Lets add it to the df
    aux['id_state'] = seq_s
    #Concatenate in max_tem
    avg_temp = pd.concat([avg_temp, aux])
```

Figure 10: For loop to concatenate the excel files

	state	average_temperature	year	id_state
0	AGUASCALIENTES	17.714283	2000	1
1	BAJA CALIFORNIA	18.810071	2000	2
2	BAJA CALIFORNIA SUR	22.146658	2000	3
3	CAMPECHE	25.879604	2000	4
6	CHIAPAS	23.534907	2000	5
...	...	...	...	...
27	TAMAULIPAS	25.100000	2019	28
28	TLAXCALA	15.800000	2019	29
29	VERACRUZ	23.300000	2019	30
30	YUCATÁN	27.300000	2019	31
31	ZACATECAS	18.300000	2019	32

640 rows × 4 columns

Figure 11: Dataframe of the result, which is stored as a CSV file

These four separated CSV files have been loaded into a jupyter notebook and have been merged by using pandas. An example of the merge is visible in figure 12. The state column has been dropped, as this info can be retrieved via the States table. The final dataframe/table for weather is visible in figure 13. This table has been stored as a CSV file.

```
#Lets merge in a big df
#Merge rain and mean temperature
aux = pd.merge(rain_df, temp_mean_df, how="outer", on=["state", "year", "id_state"])

aux
```

	state	precipitation	year	id_state	average_temperature
0	AGUASCALIENTES	417.684444	2000	1	17.714283
1	BAJA CALIFORNIA	95.783888	2000	2	18.810071
2	BAJA CALIFORNIA SUR	105.272526	2000	3	22.146658

Figure 12: Example of the merge of the separated dataframes

	id_state	year	precipitation	average_temperature	minimum_temperature	maximum_temperature
0	1	2000	417.684444	17.714283	8.496315	26.932250
1	2	2000	95.783888	18.810071	10.687024	26.933117
2	3	2000	105.272526	22.146658	13.695902	30.597415
3	4	2000	1406.374750	25.879604	19.092887	32.666320
4	5	2000	2170.151846	23.534907	17.040603	30.029211
...	...	...	...	...	...	...
635	28	2019	579.100000	25.100000	18.900000	31.200000
636	29	2019	564.200000	15.800000	7.400000	24.300000
637	30	2019	1107.600000	23.300000	17.500000	29.100000
638	31	2019	1034.200000	27.300000	20.800000	33.800000
639	32	2019	415.300000	18.300000	9.700000	27.000000

640 rows × 6 columns

Figure 13: Final weather table

### 3. Load

In order to visualize how the data is going to be related a data base diagram is made by using QuickDB. To load all the data retrieved a SQL database is built by using PostgreSQL. This will be explained in this chapter.

#### 3.1 Relationship diagram

After cleaning the databases a relationship diagram has been created by using QuickDB. How all the tables and columns are related is visible in the diagram of figure 14. Because the data of the Gross Domestic Product and the Weather is limited on state level, all the data will be connected on id\_state.

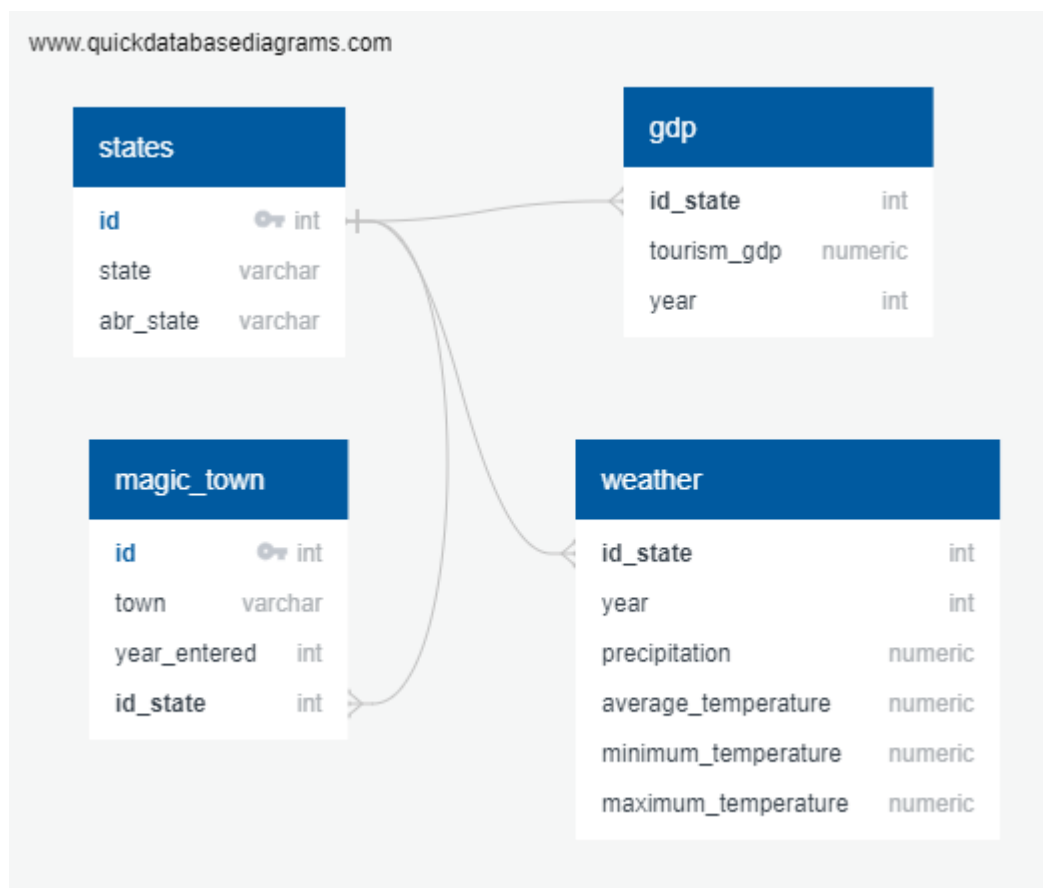


Figure 14: Relationship diagram of the database

#### 3.2 Load data into database

Once relationship diagram has been visualized a PostgreSQL database has been created. This has been done through the export to PostgreSQL function of QuickDB. In figure 15 is a part of this code shown.

Query Editor    Query History

```

6  CREATE TABLE "magic_town" (
7      "id" int NOT NULL,
8      "town" varchar NOT NULL,
9      "year_entered" int NOT NULL,
10     "id_state" int NOT NULL,
11     CONSTRAINT "pk_magic_town" PRIMARY KEY (
12         "id"
13     )
14 );
15
16 CREATE TABLE "states" (
17     "id" int NOT NULL,
18     "state" varchar NOT NULL,
19     "abr_state" varchar NOT NULL,
20     CONSTRAINT "pk_states" PRIMARY KEY (
21         "id"
22     )
23 );
24
25 CREATE TABLE "gdp" (
26     "id_state" int NOT NULL,
27     "tourism_gdp" numeric NOT NULL,
28     "year" int NOT NULL
29 );

```

Figure 15: Creating tables into the SQL database

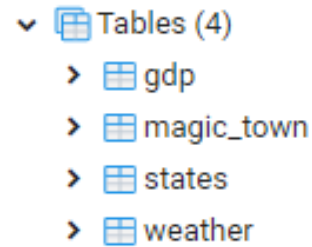


Figure 16: Tables in SQL database

After running this code all four tables have been created. In figure 16 an overview of the names of the tables in the SQL database can be found.

After creating the tables the data has to be loaded into the database. This has been done by importing the generated CSV files of the Transform part of this project.

## 4. Queries

Three queries have been created to analyze the data in the created database. First a query to calculate the average rainfall per Magic Town. The second query identifies Average State GDP per Magic Town and the last query calculates the amount of Magic Towns per state. Because of installing problems Psycopg2 in Python it was not possible to run the queries in Python and therefore, the queries have been made in Postgres, but the queries in Postgres do prove that the database is constructed and related well.

The SQL database have been loaded earlier into dataframes through queries, but at a different user. A screenshot of this step can be found in figure 16. The queries are explained in the following chapters.

```
#Lets build the database url under the scheme:  
#"postgres+psycopg2://<USERNAME>:<PASSWORD>@<IP_ADDRESS>:<PORT>/<DATABASE_NAME>"  
DATABASE_URI = f"postgres+psycopg2://{user_name}:{pwd}@localhost:5432/magical_towns_db"
```

```
#Lets import the class create_engine from sqlalchemy  
from sqlalchemy import create_engine  
engine = create_engine(DATABASE_URI)
```

```
#Lets build the connection  
conn = engine.connect()
```

```
#Lets import the states table into a pandas df  
states_db = pd.read_sql("SELECT * FROM states_db", conn)  
states_db.head()
```

	abr_state	state	id_state
0	AGS	AGUASCALIENTES	1
1	BC	BAJA CALIFORNIA	2
2	BCS	BAJA CALIFORNIA SUR	3
3	CAMP	CAMPECHE	4
4	CHIS	CHIAPAS	5

Figure 16: Loading the tables from database into dataframes in notebook

### 4.1 Average rainfall per Magic Town

A query to calculate the average rainfall per Magic Town has been made in SQL. Unfortunately, the data was only available on state level and therefore, the results are on state level. The query and the results can be found in figure 17 on the next page.

```

1 Select
2 magic_town.town,
3 states.state,
4 AVG(weather.precipitation)::NUMERIC(1000,2)
5 From states
6 Inner Join weather on weather.id_state = states.id
7 Inner Join magic_town on weather.id_state = magic_town.id_state
8 GROUP BY magic_town.town, states.state;|

```

Data Output Explain Messages Notifications

	town character varying	state character varying	avg numeric (1000,2)
1	Orizaba	VERACRUZ	1621.49
2	Coatepec	VERACRUZ	1621.49
3	Tecate	BAJA CALIFORNIA	187.75
4	Cuetzalan del Progreso	PUEBLA	1346.05
5	Huasca de Ocampo	HIDALGO	693.83
6	Mier	TAMAULIPAS	803.77
7	Zozocolco	VERACRUZ	1621.49
8	Tlaxco	TLAXCALA	720.52
9	El Rosario	SINALOA	721.91
10	San Pablo Villa de Mitla	OAXACA	1379.75

Figure 17: Average rainfall per Magic Town

## 4.2 Average State GDP per Magic Town

The second query calculates the average state GDP per Magic Town. Unfortunately, the data was only available on state level and therefore, the results are on state level. The query and the results can be found in figure 18 on the next page.

Query Editor		Query History	
1	<b>Select</b>		
2	magic_town.town,		
3	states.state,		
4	AVG(gdp.tourism_gdp)::NUMERIC(1000,2)		
5	<b>From</b> states		
6	<b>Inner Join</b> gdp <b>on</b> gdp.id_state = states.id		
7	<b>Inner Join</b> magic_town <b>on</b> gdp.id_state = magic_town.id_state		
8	<b>GROUP BY</b> magic_town.town, states.state;		

Data Output		Explain	Messages	Notifications
	town character varying	state character varying	avg numeric (1000,2)	
1	Orizaba	VERACRUZ	13844.52	
2	Coatepec	VERACRUZ	13844.52	
3	Tecate	BAJA CALIFORNIA	11598.89	
4	Cuetzalan del Progreso	PUEBLA	7953.78	
5	Huasca de Ocampo	HIDALGO	3328.84	
6	Mier	TAMAULIPAS	6479.85	
7	Zozocolco	VERACRUZ	13844.52	
8	Tlaxco	TLAXCALA	1736.55	
9	El Rosario	SINALOA	9656.73	
10	San Pablo Villa de Mitla	OAXACA	6967.14	

Figure 18: Average State GDP per Magic Town

### 4.3 Magic Towns per state

An easy query has been made to identify the amount of magic towns per state. This can be found in figure 19.

Query Editor		Query History	
1	<b>Select</b>		
2	states.state,		
3	count(magic_town.town)		
4	<b>From</b> magic_town		
5	<b>Inner Join</b> states <b>on</b> states.id = magic_town.id_state		
6	<b>GROUP BY</b> states.state		
7	<b>ORDER BY</b> states.state;		

Data Output		Explain	Messages	Notifications
	state character varying	count bigint		
1	AGUASCALIENTES	3		
2	BAJA CALIFORNIA	1		
3	BAJA CALIFORNIA SUR	2		
4	CAMPECHE	1		
5	CHIAPAS	4		
6	CHIHUAHUA	3		
7	COAHUILA	6		
8	COLIMA	1		
9	DURANGO	1		
10	ESTADO DE MEXICO	9		

Figure 19: Amount of Magic Towns per state