Setup

In [1]:
```
!pwd
```

```
/home/ubuntu/notebooks
```

Unzipping files

In [2]:
```
!unzip -l uk_house.zip
```

```
Archive:  uk_house.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
        0  2023-11-30 23:00   uk_house/
      276  2023-11-30 23:00   __MACOSX/._uk_house
     6148  2023-12-05 20:07   uk_house/.DS_Store
      176  2023-12-05 20:07   __MACOSX/uk_house/._.DS_Store
115095977  2020-02-25 00:57   uk_house/pp-2010.csv
      176  2020-02-25 00:57   __MACOSX/uk_house/._pp-2010.csv
236396933  2020-02-25 00:55   uk_house/pp-2006.csv
      176  2020-02-25 00:55   __MACOSX/uk_house/._pp-2006.csv
227354238  2020-02-25 00:56   uk_house/pp-2007.csv
      176  2020-02-25 00:56   __MACOSX/uk_house/._pp-2007.csv
108821824  2020-02-25 00:56   uk_house/pp-2009.csv
      176  2020-02-25 00:56   __MACOSX/uk_house/._pp-2009.csv
116206943  2020-02-25 00:56   uk_house/pp-2008.csv
      176  2020-02-25 00:56   __MACOSX/uk_house/._pp-2008.csv
---------                     -------
803883395                     14 files
```

In [9]:
```
!unzip uk_house.zip
```

```
Archive:  uk_house.zip
   creating: uk_house/
  inflating: __MACOSX/._uk_house
  inflating: uk_house/.DS_Store
  inflating: __MACOSX/uk_house/._.DS_Store
  inflating: uk_house/pp-2010.csv
  inflating: __MACOSX/uk_house/._pp-2010.csv
  inflating: uk_house/pp-2006.csv
  inflating: __MACOSX/uk_house/._pp-2006.csv
  inflating: uk_house/pp-2007.csv
  inflating: __MACOSX/uk_house/._pp-2007.csv
  inflating: uk_house/pp-2009.csv
  inflating: __MACOSX/uk_house/._pp-2009.csv
  inflating: uk_house/pp-2008.csv
  inflating: __MACOSX/uk_house/._pp-2008.csv
```

As the goal of our project is to make recommendations for consumers looking to purchase housing in the present day we've chosen to limit our data to the most recent 5 years, as taking account of data from too far back may needlessly influence decisionmaking on property regions that were valuable or cheap historically,but are no longer dear

In [2]:
```
#!rm archive.zip optional
```

Unzipping the data and deleting the zip file to save storage

In [1]:
```
!find uk_house -type f -name "*.csv" -exec csvstack {} + > complete_df.cs
```

combinding to create the csvstack

In [3]:
```
#!rm -r uk_housing_prices optional
```

Removing the file to save storage

In [2]:
```
!echo "UID,Price,Date_of_Transfer,Postcode,Property_Type,Old_New,Duration
!cat headers.csv complete_df.csv > final_df.csv
!rm headers.csv
!rm complete_df.csv
```

creating the headers and adding them to the csv file

Basic validation and cleaning

In [3]:
```
!wc -l final_df.csv
```

```
4534440 final_df.csv
```

This matches up with the size of the data we're expecting.

In [4]:
```
!head final_df.csv
```

```
UID,Price,Date_of_Transfer,Postcode,Property_Type,Old_New,Duration,PAON,
SAON,Street,Locality,City,District,County,PPD_Category_Type,Record_Statu
s
{A9734ACB-794C-4471-9179-C7F4BC550815},115000,2006-07-28 00:00,NP20 5HL,
T,N,F,18,,LOCKE STREET,NEWPORT,NEWPORT,NEWPORT,NEWPORT,A,A
{687DDA6E-B488-4C93-A051-C7F4C48118E8},200000,2006-02-20 00:00,NE22 6PE,
D,N,F,32,,NOTTINGHAM COURT,BEDLINGTON,BEDLINGTON,WANSBECK,NORTHUMBERLAND
,A,A
{CC8EFE32-AC59-4D4B-9FF9-C7F4C7DA0E08},145000,2006-09-08 00:00,TQ4 5NE,T
,N,F,38,,CORSHAM ROAD,PAIGNTON,PAIGNTON,TORBAY,TORBAY,A,A
{4C805BC4-AE40-4663-AD5B-C7F4CD4EB18A},184800,2006-12-11 00:00,YO31 0TY,
S,N,F,41,,THIRD AVENUE,YORK,YORK,YORK,YORK,A,A
{F79A8768-CCFF-4043-8883-C7F4D553364B},56000,2006-01-19 00:00,ST10 1JT,T
,N,F,26,,FROGHALL ROAD,CHEADLE,STOKE-ON-TRENT,STAFFORDSHIRE MOORLANDS,ST
AFFORDSHIRE,A,A
{27D5915D-D763-4551-A264-C7F4DADA89BD},249950,2006-12-19 00:00,SW18 3TD,
T,N,F,74A,,WALDRON ROAD,LONDON,LONDON,WANDSWORTH,GREATER LONDON,A,A
{E7BB4EDC-AB2B-4B43-8E1D-C7F4DC0C37EF},167000,2006-09-01 00:00,FY3 9HB,S
,N,F,19,,WILKINSON AVENUE,BLACKPOOL,BLACKPOOL,BLACKPOOL,BLACKPOOL,A,A
{614A3ECE-67CB-4BBA-B70F-C7F4DD12C9E7},175000,2006-01-28 00:00,GU11 3JS,
T,N,F,36,,PEROWNE STREET,ALDERSHOT,ALDERSHOT,RUSHMOOR,HAMPSHIRE,A,A
{8ACCE855-A0F2-4E20-B971-C7F4DEA90A5E},118745,2006-08-25 00:00,PR1 1JB,F
,Y,L,BEDFORD COURT,APARTMENT 24,CRAGGS ROW,PRESTON,PRESTON,PRESTON,LANCA
SHIRE,A,A
```

from the head we can see that there are weird curly brackets around our Unique Identifier, lets remove them as they do not add to the analysis. Inspecting the data, it also doesn't actually have time down to the hour and minute, so lets remove that as well

In [5]:
```
!sed -i -e 's/[{}]//g' -e 's/ 00:00//g' /home/ubuntu/notebooks/Final-Proj
```

In [6]:
```
!head final_df.csv
```

```
UID,Price,Date_of_Transfer,Postcode,Property_Type,Old_New,Duration,PAON,
SAON,Street,Locality,City,District,County,PPD_Category_Type,Record_Statu
s
A9734ACB-794C-4471-9179-C7F4BC550815,115000,2006-07-28,NP20 5HL,T,N,F,18
,,LOCKE STREET,NEWPORT,NEWPORT,NEWPORT,NEWPORT,A,A
687DDA6E-B488-4C93-A051-C7F4C48118E8,200000,2006-02-20,NE22 6PE,D,N,F,32
,,NOTTINGHAM COURT,BEDLINGTON,BEDLINGTON,WANSBECK,NORTHUMBERLAND,A,A
CC8EFE32-AC59-4D4B-9FF9-C7F4C7DA0E08,145000,2006-09-08,TQ4 5NE,T,N,F,38,
,CORSHAM ROAD,PAIGNTON,PAIGNTON,TORBAY,TORBAY,A,A
4C805BC4-AE40-4663-AD5B-C7F4CD4EB18A,184800,2006-12-11,YO31 0TY,S,N,F,41
,,THIRD AVENUE,YORK,YORK,YORK,YORK,A,A
F79A8768-CCFF-4043-8883-C7F4D553364B,56000,2006-01-19,ST10 1JT,T,N,F,26,
,FROGHALL ROAD,CHEADLE,STOKE-ON-TRENT,STAFFORDSHIRE MOORLANDS,STAFFORDSH
IRE,A,A
27D5915D-D763-4551-A264-C7F4DADA89BD,249950,2006-12-19,SW18 3TD,T,N,F,74
A,,WALDRON ROAD,LONDON,LONDON,WANDSWORTH,GREATER LONDON,A,A
E7BB4EDC-AB2B-4B43-8E1D-C7F4DC0C37EF,167000,2006-09-01,FY3 9HB,S,N,F,19,
,WILKINSON AVENUE,BLACKPOOL,BLACKPOOL,BLACKPOOL,BLACKPOOL,A,A
614A3ECE-67CB-4BBA-B70F-C7F4DD12C9E7,175000,2006-01-28,GU11 3JS,T,N,F,36
,,PEROWNE STREET,ALDERSHOT,ALDERSHOT,RUSHMOOR,HAMPSHIRE,A,A
8ACCE855-A0F2-4E20-B971-C7F4DEA90A5E,118745,2006-08-25,PR1 1JB,F,Y,L,BED
FORD COURT,APARTMENT 24,CRAGGS ROW,PRESTON,PRESTON,PRESTON,LANCASHIRE,A,
A
```

PostgreSQL Setup

In [3]:
```
!pip freeze | grep -E 'ipython-sql|psycopg2'
```

```
ipython-sql==0.4.1
psycopg2==2.9.5
psycopg2-binary==2.9.5
```

In [4]:
```
%load_ext sql
```

In [9]:
```
!dropdb -U student FinalProj
```

```
dropdb: error: database removal failed: ERROR:  database "FinalProj" doe
s not exist
```

In [10]:
```
!createdb -U student FinalProj
```

In [5]:
```
%sql postgresql://student@/FinalProj
```

In [13]:
```
!head -n 10 final_df.csv | csvstat
```

```
/home/ubuntu/.local/lib/python3.8/site-packages/agate/table/from_csv.py:
74: RuntimeWarning: Error sniffing CSV dialect: Could not determine deli
miter
  1. "UID"

        Type of data:          Text
        Contains null values:  False
```

Unique values:	9
Longest value:	36 characters
Most common values:	A9734ACB-794C-4471-9179-C7F4BC550815 (1x) 687DDA6E-B488-4C93-A051-C7F4C48118E8 (1x) CC8EFE32-AC59-4D4B-9FF9-C7F4C7DA0E08 (1x) 4C805BC4-AE40-4663-AD5B-C7F4CD4EB18A (1x) F79A8768-CCFF-4043-8883-C7F4D553364B (1x)

2. "Price"

Type of data:	Number
Contains null values:	False
Unique values:	9
Smallest value:	56000
Largest value:	249950
Sum:	1411495
Mean:	156832.778
Median:	167000
StDev:	56189.118
Most common values:	115000 (1x) 200000 (1x) 145000 (1x) 184800 (1x) 56000 (1x)

3. "Date_of_Transfer"

Type of data:	Date
Contains null values:	False
Unique values:	9
Smallest value:	2006-01-19
Largest value:	2006-12-19
Most common values:	2006-07-28 (1x) 2006-02-20 (1x) 2006-09-08 (1x) 2006-12-11 (1x) 2006-01-19 (1x)

4. "Postcode"

Type of data:	Text
Contains null values:	False
Unique values:	9
Longest value:	8 characters
Most common values:	NP20 5HL (1x) NE22 6PE (1x) TQ4 5NE (1x) YO31 0TY (1x) ST10 1JT (1x)

5. "Property_Type"

Type of data:	Text

```
      Contains null values:  False
      Unique values:         4
      Longest value:         1 characters
      Most common values:    T (5x)
                             S (2x)
                             D (1x)
                             F (1x)
```

6. "Old_New"

```
      Type of data:          Boolean
      Contains null values:  False
      Unique values:         2
      Most common values:    False (8x)
                             True (1x)
```

7. "Duration"

```
      Type of data:          Text
      Contains null values:  False
      Unique values:         2
      Longest value:         1 characters
      Most common values:    F (8x)
                             L (1x)
```

8. "PAON"

```
      Type of data:          Text
      Contains null values:  False
      Unique values:         9
      Longest value:         13 characters
      Most common values:    18 (1x)
                             32 (1x)
                             38 (1x)
                             41 (1x)
                             26 (1x)
```

9. "SAON"

```
      Type of data:          Text
      Contains null values:  True (excluded from calculations)
      Unique values:         2
      Longest value:         12 characters
      Most common values:    None (8x)
                             APARTMENT 24 (1x)
```

10. "Street"

```
      Type of data:          Text
      Contains null values:  False
      Unique values:         9
      Longest value:         16 characters
      Most common values:    LOCKE STREET (1x)
```

```
                                NOTTINGHAM COURT (1x)
                                CORSHAM ROAD (1x)
                                THIRD AVENUE (1x)
                                FROGHALL ROAD (1x)
```

11. "Locality"

```
       Type of data:          Text
       Contains null values:  False
       Unique values:         9
       Longest value:         10 characters
       Most common values:    NEWPORT (1x)
                              BEDLINGTON (1x)
                              PAIGNTON (1x)
                              YORK (1x)
                              CHEADLE (1x)
```

12. "City"

```
       Type of data:          Text
       Contains null values:  False
       Unique values:         9
       Longest value:         14 characters
       Most common values:    NEWPORT (1x)
                              BEDLINGTON (1x)
                              PAIGNTON (1x)
                              YORK (1x)
                              STOKE-ON-TRENT (1x)
```

13. "District"

```
       Type of data:          Text
       Contains null values:  False
       Unique values:         9
       Longest value:         23 characters
       Most common values:    NEWPORT (1x)
                              WANSBECK (1x)
                              TORBAY (1x)
                              YORK (1x)
                              STAFFORDSHIRE MOORLANDS (1x)
```

14. "County"

```
       Type of data:          Text
       Contains null values:  False
       Unique values:         9
       Longest value:         14 characters
       Most common values:    NEWPORT (1x)
                              NORTHUMBERLAND (1x)
                              TORBAY (1x)
                              YORK (1x)
                              STAFFORDSHIRE (1x)
```

15. "PPD_Category_Type"

        Type of data: Text
        Contains null values: False
        Unique values: 1
        Longest value: 1 characters
        Most common values: A (9x)

16. "Record_Status"

        Type of data: Text
        Contains null values: False
        Unique values: 1
        Longest value: 1 characters
        Most common values: A (9x)

Row count: 9

Creating and populating tables and Further Cleaning

In [14]:
```
%%sql

DROP TABLE IF EXISTS property_sales;

CREATE TABLE property_sales (
    TransactionUniqueIdentifier VARCHAR(36) NOT NULL,
    Price NUMERIC NOT NULL,
    DateOfTransfer DATE NOT NULL,
    Postcode VARCHAR(8),
    PropertyType CHAR(1) NOT NULL,
    OldNew CHAR(1) NOT NULL,
    Duration CHAR(1),
    PAON VARCHAR,
    SAON VARCHAR,
    Street VARCHAR,
    Locality VARCHAR,
    TownCity VARCHAR NOT NULL,
    District VARCHAR,
    County VARCHAR NOT NULL,
    PPDCategoryType CHAR(1) NOT NULL,
    RecordStatus CHAR(1) NOT NULL
);
```

 * postgresql://student@/FinalProj
Done.
Done.

Out[14]: []

In [15]:
```
%%sql
COMMENT ON COLUMN property_sales.TransactionUniqueIdentifier IS 'A unique
COMMENT ON COLUMN property_sales.Price IS 'The sale price stated on the t
COMMENT ON COLUMN property_sales.DateOfTransfer IS 'The completion date o
COMMENT ON COLUMN property_sales.Postcode IS 'The postal code in use at t
COMMENT ON COLUMN property_sales.PropertyType IS 'Type of property (D = D
COMMENT ON COLUMN property_sales.OldNew IS 'Indicates if the property is
COMMENT ON COLUMN property_sales.Duration IS 'Tenure of the property (F =
COMMENT ON COLUMN property_sales.PAON IS 'Primary Addressable Object Name
COMMENT ON COLUMN property_sales.SAON IS 'Secondary Addressable Object Na
COMMENT ON COLUMN property_sales.Street IS 'The street name of the proper
COMMENT ON COLUMN property_sales.Locality IS 'The locality or area where
COMMENT ON COLUMN property_sales.TownCity IS 'The town or city where the
COMMENT ON COLUMN property_sales.District IS 'The district where the prop
COMMENT ON COLUMN property_sales.County IS 'The county where the property
COMMENT ON COLUMN property_sales.PPDCategoryType IS 'Indicates the type o
COMMENT ON COLUMN property_sales.RecordStatus IS 'Status of the records -
```

```
 * postgresql://student@/FinalProj
Done.
Done.
Done.
Done.
Done.
Done.
Done.
Done.
Done.
Done.
Done.
Done.
Done.
Done.
Done.
Done.
```

Out[15]: []

In [16]:
```
!pwd
```

```
/home/ubuntu/notebooks/Final-Project
```

In [17]:
```
%%sql
COPY property_sales FROM '/home/ubuntu/notebooks/Final-Project/final_df.c
CSV
HEADER;
```

```
 * postgresql://student@/FinalProj
4534439 rows affected.
```

Out[17]: []

In [18]:
```
%%sql
SELECT COUNT(*) FROM property_sales;
```

```
 * postgresql://student@/FinalProj
1 rows affected.
```

Out[18]:

count
4534439

In [19]:
```
%%sql
SELECT * FROM property_sales
LIMIT 10
```

```
 * postgresql://student@/FinalProj
10 rows affected.
```

Out[19]:

transactionuniqueidentifier	price	dateoftransfer	postcode	propertytype	oldnew	dur
A9734ACB-794C-4471-9179-C7F4BC550815	115000	2006-07-28	NP20 5HL	T	N	
687DDA6E-B488-4C93-A051-C7F4C48118E8	200000	2006-02-20	NE22 6PE	D	N	
CC8EFE32-AC59-4D4B-9FF9-C7F4C7DA0E08	145000	2006-09-08	TQ4 5NE	T	N	
4C805BC4-AE40-4663-AD5B-C7F4CD4EB18A	184800	2006-12-11	YO31 0TY	S	N	
F79A8768-CCFF-4043-8883-C7F4D553364B	56000	2006-01-19	ST10 1JT	T	N	
27D5915D-D763-4551-A264-C7F4DADA89BD	249950	2006-12-19	SW18 3TD	T	N	
E7BB4EDC-AB2B-4B43-8E1D-C7F4DC0C37EF	167000	2006-09-01	FY3 9HB	S	N	
614A3ECE-67CB-4BBA-B70F-C7F4DD12C9E7	175000	2006-01-28	GU11 3JS	T	N	
8ACCE855-A0F2-4E20-B971-C7F4DEA90A5E	118745	2006-08-25	PR1 1JB	F	Y	
FA8746B5-1D89-489F-946D-C7F4E1A49406	215950	2006-11-24	TN9 1QU	F	Y	

let's combine the PAON, SAON, and street to create a more unique specific address, furthermore, given we already have City, District, and County let's remove Locality as a column, we dont really need this much geographical data, and if needed, the Postcode is a much more accurate measure of the geographical region the house is located in

In [20]:
```
%%sql

ALTER TABLE property_sales
DROP COLUMN PAON,
DROP COLUMN SAON,
DROP COLUMN Street,
DROP COLUMN postcode;
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[20]: []

In [21]:
```
%%sql
ALTER TABLE property_sales DROP COLUMN Locality;
ALTER TABLE property_sales DROP COLUMN TransactionUniqueIdentifier;
```

```
 * postgresql://student@/FinalProj
Done.
Done.
```

Out[21]: []

let's also remove the recordstatus column as it is not relevant for analysis

In [22]:
```
%%sql
ALTER TABLE property_sales DROP COLUMN recordstatus;
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[22]: []

In [23]:
```
%%sql
SELECT * FROM property_sales
LIMIT 10
```

```
 * postgresql://student@/FinalProj
10 rows affected.
```

Out[23]:

price	dateoftransfer	propertytype	oldnew	duration	towncity	district
115000	2006-07-28	T	N	F	NEWPORT	NEWPORT
200000	2006-02-20	D	N	F	BEDLINGTON	WANSBECK
145000	2006-09-08	T	N	F	PAIGNTON	TORBAY
184800	2006-12-11	S	N	F	YORK	YORK
56000	2006-01-19	T	N	F	STOKE-ON-TRENT	STAFFORDSHIRE MOORLANDS
249950	2006-12-19	T	N	F	LONDON	WANDSWORTH
167000	2006-09-01	S	N	F	BLACKPOOL	BLACKPOOL
175000	2006-01-28	T	N	F	ALDERSHOT	RUSHMOOR
118745	2006-08-25	F	Y	L	PRESTON	PRESTON
215950	2006-11-24	F	Y	L	TONBRIDGE	TONBRIDGE AND MALLING

Dealing with NULL values

In [24]:
```
%%sql
SELECT
    COUNT(*) - COUNT(Price) AS Null_Price,
    COUNT(*) - COUNT(DateOfTransfer) AS Null_DateOfTransfer,
    COUNT(*) - COUNT(PropertyType) AS Null_PropertyType,
    COUNT(*) - COUNT(OldNew) AS Null_OldNew,
    COUNT(*) - COUNT(Duration) AS Null_Duration,
    COUNT(*) - COUNT(TownCity) AS Null_TownCity,
    COUNT(*) - COUNT(District) AS Null_District,
    COUNT(*) - COUNT(County) AS Null_County,
    COUNT(*) - COUNT(PPDCategoryType) AS Null_PPDCategoryType
FROM property_sales;
```

```
 * postgresql://student@/FinalProj
1 rows affected.
```

Out[24]:

null_price	null_dateoftransfer	null_propertytype	null_oldnew	null_duration	null_townci
0	0	0	0	0	

Adding dimensionality to the data

Looking at our data, we've decided to add two primary dimensions

1.Date dimension, having the dateoftransfer for all values allows a significant amount of time dimensionality, and should be taken advantage of

2.Location dimension, let's dimensionalize the location for easier querying later

Creating the date dimension

In [25]:
```
%%sql
DROP TABLE IF EXISTS date_dimension
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[25]: []

In [26]:
```
%%sql

CREATE TABLE date_dimension (
    date_key SERIAL PRIMARY KEY,
    actual_date DATE UNIQUE,
    year INTEGER,
    month INTEGER,
    day INTEGER,
    quarter INTEGER,
    week_of_year INTEGER
);
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[26]: []

In [27]:
```
%%sql
COMMENT ON COLUMN date_dimension.date_key IS 'The primary key representin
COMMENT ON COLUMN date_dimension.year IS 'The year portion of the date';
COMMENT ON COLUMN date_dimension.month IS 'The month portion of the date'
COMMENT ON COLUMN date_dimension.day IS 'The day portion of the date';
COMMENT ON COLUMN date_dimension.quarter IS 'The quarter of the year corr
COMMENT ON COLUMN date_dimension.week_of_year IS 'The week of the year co
```

```
 * postgresql://student@/FinalProj
Done.
Done.
Done.
Done.
Done.
Done.
```

Out[27]: []

populating the new dimension with data from dateoftransfer

In [28]:
```
%%sql
INSERT INTO date_dimension (actual_date, year, month, day, quarter, week_
SELECT DISTINCT
    DateOfTransfer,
    EXTRACT(YEAR FROM DateOfTransfer) AS year,
    EXTRACT(MONTH FROM DateOfTransfer) AS month,
    EXTRACT(DAY FROM DateOfTransfer) AS day,
    EXTRACT(QUARTER FROM DateOfTransfer) AS quarter,
    EXTRACT(WEEK FROM DateOfTransfer) AS week_of_year
FROM property_sales;
```

```
 * postgresql://student@/FinalProj
1826 rows affected.
```

Out[28]: []

In [29]:
```
%%sql
SELECT * FROM date_dimension LIMIT 10;
```

```
 * postgresql://student@/FinalProj
10 rows affected.
```

Out[29]:

date_key	actual_date	year	month	day	quarter	week_of_year
1	2006-07-31	2006	7	31	3	31
2	2008-07-14	2008	7	14	3	29
3	2008-04-03	2008	4	3	2	14
4	2006-12-22	2006	12	22	4	51
5	2009-12-20	2009	12	20	4	51
6	2008-07-31	2008	7	31	3	31
7	2008-12-18	2008	12	18	4	51
8	2006-08-17	2006	8	17	3	33
9	2009-06-25	2009	6	25	2	26
10	2010-01-21	2010	1	21	1	3

adding date_id as a key back to the original table

In [30]:
```
%%sql
ALTER TABLE property_sales
ADD COLUMN date_key INTEGER,
ADD CONSTRAINT fk_date
    FOREIGN KEY (date_key)
    REFERENCES date_dimension(date_key);
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[30]: []

In [31]:
```
%%sql
SELECT * FROM property_sales LIMIT 10;
```

```
 * postgresql://student@/FinalProj
10 rows affected.
```

Out[31]:

price	dateoftransfer	propertytype	oldnew	duration	towncity	district
115000	2006-07-28	T	N	F	NEWPORT	NEWPORT
200000	2006-02-20	D	N	F	BEDLINGTON	WANSBECK
145000	2006-09-08	T	N	F	PAIGNTON	TORBAY
184800	2006-12-11	S	N	F	YORK	YORK
56000	2006-01-19	T	N	F	STOKE-ON-TRENT	STAFFORDSHIRE MOORLANDS
249950	2006-12-19	T	N	F	LONDON	WANDSWORTH
167000	2006-09-01	S	N	F	BLACKPOOL	BLACKPOOL
175000	2006-01-28	T	N	F	ALDERSHOT	RUSHMOOR
118745	2006-08-25	F	Y	L	PRESTON	PRESTON
215950	2006-11-24	F	Y	L	TONBRIDGE	TONBRIDGE AND MALLING

In [32]:
```
%%sql
UPDATE property_sales
SET date_key = (SELECT date_key FROM date_dimension WHERE actual_date = D
```

```
 * postgresql://student@/FinalProj
4534439 rows affected.
```

Out[32]: []

In [33]:
```
%%sql
SELECT
    ps.*,
    dd.*
FROM property_sales ps
JOIN date_dimension dd ON ps.date_key = dd.date_key
WHERE dd.quarter = 3 AND dd.year = 2008
LIMIT 12;
```

```
 * postgresql://student@/FinalProj
12 rows affected.
```

Out[33]:

price	dateoftransfer	propertytype	oldnew	duration	towncity	district
190000	2008-08-22	T	N	F	GRAYS	THURROCK
100000	2008-07-22	F	Y	L	SOUTHAMPTON	NEW FOREST
450000	2008-08-05	D	N	F	KINGSBRIDGE	SOUTH HAMS
180000	2008-07-24	T	N	F	LONDON	SOUTHWARK
150000	2008-07-11	T	N	F	CARDIFF	CARDIFF
415000	2008-08-07	F	N	L	LONDON	CITY OF WESTMINSTER
110000	2008-08-19	S	N	F	BOSTON	BOSTON
163500	2008-09-04	S	N	F	YORK	YORK
185000	2008-07-30	F	Y	L	HIGH WYCOMBE	WYCOMBE
214750	2008-09-17	F	N	L	LONDON	LEWISHAM
643500	2008-08-08	D	N	F	BARNET	BARNET
60000	2008-09-29	S	N	F	BIRMINGHAM	BIRMINGHAM

after successfully verifying the new dimension is working, we can drop the original dateoftransfer table

In [34]:
```
%%sql
ALTER TABLE property_sales
DROP COLUMN dateoftransfer;
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[34]: []

In [35]:
```
%%sql
SELECT * FROM property_sales
LIMIT 5
```

```
 * postgresql://student@/FinalProj
5 rows affected.
```

Out[35]:

price	propertytype	oldnew	duration	towncity	district	county
450000	D	N	F	ROCHFORD	ROCHFORD	ESSEX
214750	F	N	L	LONDON	LEWISHAM	GREATER LONDON
195000	T	N	F	HECKMONDWIKE	KIRKLEES	WEST YORKSHIRE
305000	D	N	F	NEWNHAM	FOREST OF DEAN	GLOUCESTERSHIRE
86000	F	N	L	SWANSEA	SWANSEA	SWANSEA

Creating City Dimension

In [36]:
```
%%sql
DROP TABLE IF EXISTS county_dimension CASCADE;
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[36]: []

In [37]:
```
%%sql

CREATE TABLE county_dimension (
    county_key SERIAL PRIMARY KEY,
    county VARCHAR
);
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[37]: []

In [38]:
```
%%sql
COMMENT ON COLUMN county_dimension.county_key IS 'The primary key represe
COMMENT ON COLUMN county_dimension.county IS 'The county the property is
```

```
 * postgresql://student@/FinalProj
Done.
Done.
```

Out[38]: []

In [39]:
```
%%sql
INSERT INTO county_dimension(county)
SELECT DISTINCT
    county
FROM property_sales
```

```
 * postgresql://student@/FinalProj
118 rows affected.
```

Out[39]: []

In [40]:
```
%%sql
SELECT * FROM county_dimension
LIMIT 10
```

```
 * postgresql://student@/FinalProj
10 rows affected.
```

Out[40]:

county_key	county
1	SOUTHAMPTON
2	BEDFORD
3	BLAENAU GWENT
4	COUNTY DURHAM
5	PEMBROKESHIRE
6	KENT
7	CONWY
8	MERTHYR TYDFIL
9	SOUTHEND-ON-SEA
10	WOKINGHAM

In [41]:
```
%%sql
ALTER TABLE property_sales
ADD COLUMN county_key INTEGER,
ADD CONSTRAINT fk_county
    FOREIGN KEY (county_key)
    REFERENCES county_dimension(county_key);
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[41]: []

In [42]:
```
%%sql
SELECT * FROM property_sales LIMIT 10;
```

```
 * postgresql://student@/FinalProj
10 rows affected.
```

Out[42]:

price	propertytype	oldnew	duration	towncity	district	coun
450000	D	N	F	ROCHFORD	ROCHFORD	ESSI
214750	F	N	L	LONDON	LEWISHAM	GREATER LONDC
195000	T	N	F	HECKMONDWIKE	KIRKLEES	WEST YORKSHI
305000	D	N	F	NEWNHAM	FOREST OF DEAN	GLOUCESTERSHI
86000	F	N	L	SWANSEA	SWANSEA	SWANSI
643500	D	N	F	BARNET	BARNET	GREATER LONDC
60000	S	N	F	BIRMINGHAM	BIRMINGHAM	WEST MIDLAN
86000	F	N	L	BLANDFORD FORUM	NORTH DORSET	DORSI
109995	F	Y	L	SWANSEA	SWANSEA	SWANSI
123000	F	N	L	YORK	YORK	YOI

In [43]:
```
%%sql
UPDATE property_sales
SET county_key = (
    SELECT cd.county_key
    FROM county_dimension cd
    WHERE cd.county = property_sales.county
);
```

```
 * postgresql://student@/FinalProj
4534439 rows affected.
```

Out[43]: []

In [44]:
```
%%sql
SELECT * FROM property_sales LIMIT 10;
```

```
 * postgresql://student@/FinalProj
10 rows affected.
```

Out[44]:

price	propertytype	oldnew	duration	towncity	district	county	ppd
123000	F	N	L	YORK	YORK	YORK	
91178	T	Y	F	CHORLEY	CHORLEY	LANCASHIRE	
97895	S	N	F	PONTEFRACT	WAKEFIELD	WEST YORKSHIRE	
114000	S	N	F	NEWTON-LE-WILLOWS	ST HELENS	MERSEYSIDE	
249995	T	N	F	LEEDS	LEEDS	WEST YORKSHIRE	
195000	S	N	F	DEAL	DOVER	KENT	
260000	F	Y	L	TADWORTH	REIGATE AND BANSTEAD	SURREY	
81500	S	N	L	MANCHESTER	WIGAN	GREATER MANCHESTER	
65000	T	N	F	ROCHESTER	MEDWAY	MEDWAY	
175000	D	N	F	ELLESMERE	SHROPSHIRE	SHROPSHIRE	

In [45]:

```
%%sql
SELECT *
FROM property_sales
JOIN county_dimension ON property_sales.county_key = county_dimension.cou
LIMIT 12;
```

```
 * postgresql://student@/FinalProj
12 rows affected.
```

Out[45]:

price	propertytype	oldnew	duration	towncity	district	
195000	S	N	F	DEAL	DOVER	
260000	F	Y	L	TADWORTH	REIGATE AND BANSTEAD	
81500	S	N	L	MANCHESTER	WIGAN	M
65000	T	N	F	ROCHESTER	MEDWAY	
175000	D	N	F	ELLESMERE	SHROPSHIRE	
111155	S	N	F	SWINDON	SWINDON	
159000	T	N	F	LEICESTER	LEICESTER	
63000	T	N	L	IPSWICH	IPSWICH	
111000	S	N	F	RUSHDEN	EAST NORTHAMPTONSHIRE	NORTHA
198000	T	N	F	BASILDON	BASILDON	
175000	T	N	F	RYE	ROTHER	E
137500	T	N	F	PETERBOROUGH	FENLAND	CAM

In [46]:
```
%%sql
ALTER TABLE property_sales
DROP COLUMN county;
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[46]:
```
[]
```

In [47]:
```
%%sql
SELECT * FROM property_sales
LIMIT 5
```

```
 * postgresql://student@/FinalProj
5 rows affected.
```

Out[47]:

price	propertytype	oldnew	duration	towncity	district	ppdcategorytype	da
175000	T	N	F	RYE	ROTHER	A	
137500	T	N	F	PETERBOROUGH	FENLAND	A	
247500	S	N	F	TOTNES	SOUTH HAMS	A	
80000	T	N	F	LEEDS	LEEDS	A	
128000	T	N	L	SOUTHPORT	SEFTON	A	

Creating the district_towncity dimention

In [48]:
```
%%sql
DROP TABLE IF EXISTS district_dimension CASCADE;
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[48]: []

In [49]:
```
%%sql

CREATE TABLE district_dimension (
    district_key SERIAL PRIMARY KEY,
    district VARCHAR
);
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[49]: []

In [50]:
```
%%sql
COMMENT ON COLUMN district_dimension.district_key IS 'The primary key rep
COMMENT ON COLUMN district_dimension.district IS 'The district the proper
```

```
 * postgresql://student@/FinalProj
Done.
Done.
```

Out[50]: []

In [51]:
```
%%sql
INSERT INTO district_dimension(district)
SELECT DISTINCT
    district
FROM property_sales
```

```
 * postgresql://student@/FinalProj
396 rows affected.
```

Out[51]: []

In [52]:
```
%%sql
SELECT * FROM district_dimension
LIMIT 10
```

```
 * postgresql://student@/FinalProj
10 rows affected.
```

Out[52]:

district_key	district
1	BEDFORD
2	DONCASTER
3	OADBY AND WIGSTON
4	CONWY
5	RUGBY
6	MERTHYR TYDFIL
7	STAFFORDSHIRE MOORLANDS
8	NORTH EAST DERBYSHIRE
9	EAST DEVON
10	GATESHEAD

In [53]:
```
%%sql
ALTER TABLE property_sales
ADD COLUMN district_key INTEGER,
ADD CONSTRAINT fk_district
    FOREIGN KEY (district_key)
    REFERENCES district_dimension(district_key);
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[53]: []

In [54]:
```
%%sql
UPDATE property_sales
SET district_key = (
    SELECT dd.district_key
    FROM district_dimension dd
    WHERE dd.district = property_sales.district
);
```

```
 * postgresql://student@/FinalProj
4534439 rows affected.
```

Out[54]: []

In [55]:
```
%%sql
SELECT * FROM property_sales
LIMIT 5
```

```
 * postgresql://student@/FinalProj
5 rows affected.
```

Out[55]:

price	propertytype	oldnew	duration	towncity	district	ppdcategorytype	dat
247500	S	N	F	TOTNES	SOUTH HAMS	A	
220000	D	N	F	YEOVIL	SOUTH SOMERSET	A	
133000	T	N	F	EVESHAM	WYCHAVON	A	
40000	T	N	F	WEST BROMWICH	SANDWELL	A	
745000	D	N	F	HARPENDEN	ST ALBANS	A	

In [56]:
```
%%sql
SELECT *
FROM property_sales
JOIN district_dimension ON property_sales.district_key = district_dimensi
LIMIT 12;
```

```
 * postgresql://student@/FinalProj
12 rows affected.
```

Out[56]:

price	propertytype	oldnew	duration	towncity	district	ppdcategorytype	dat
247500	S	N	F	TOTNES	SOUTH HAMS	A	
220000	D	N	F	YEOVIL	SOUTH SOMERSET	A	
133000	T	N	F	EVESHAM	WYCHAVON	A	
40000	T	N	F	WEST BROMWICH	SANDWELL	A	
745000	D	N	F	HARPENDEN	ST ALBANS	A	
128500	S	N	F	HALESOWEN	DUDLEY	A	
125000	S	N	F	FARINGDON	VALE OF WHITE HORSE	A	
219000	F	N	L	EAST MOLESEY	ELMBRIDGE	A	
86500	T	N	F	LINCOLN	LINCOLN	A	
105000	S	N	F	SLEAFORD	NORTH KESTEVEN	A	
117500	T	N	F	WALLASEY	WIRRAL	A	
499950	D	N	F	UXBRIDGE	SOUTH BUCKS	A	

In [57]:
```
%%sql
ALTER TABLE property_sales
DROP COLUMN district;
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[57]: []

In [58]:
```
%%sql
SELECT * FROM property_sales
LIMIT 5
```

```
 * postgresql://student@/FinalProj
5 rows affected.
```

Out[58]:

price	propertytype	oldnew	duration	towncity	ppdcategorytype	date_key	count
247500	S	N	F	TOTNES	A	1429	
220000	D	N	F	YEOVIL	A	1276	
133000	T	N	F	EVESHAM	A	999	
40000	T	N	F	WEST BROMWICH	A	1383	
745000	D	N	F	HARPENDEN	A	645	

Creating the towncity dimension

In [59]:
```
%%sql
DROP TABLE IF EXISTS towncity_dimension CASCADE;
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[59]: []

In [60]:
```
%%sql

CREATE TABLE towncity_dimension (
    towncity_key SERIAL PRIMARY KEY,
    towncity VARCHAR
);
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[60]: []

In [61]:
```
%%sql
COMMENT ON COLUMN towncity_dimension.towncity_key IS 'The primary key rep
COMMENT ON COLUMN towncity_dimension.towncity IS 'The towncity the proper
```

```
 * postgresql://student@/FinalProj
Done.
Done.
```

Out[61]: []

In [62]:
```
%%sql
INSERT INTO towncity_dimension(towncity)
SELECT DISTINCT
    towncity
FROM property_sales
```

```
 * postgresql://student@/FinalProj
1166 rows affected.
```

Out[62]: []

In [63]:
```
%%sql
SELECT * FROM towncity_dimension
LIMIT 10
```

```
 * postgresql://student@/FinalProj
10 rows affected.
```

Out[63]:

towncity_key	towncity
1	BRENTWOOD
2	MARAZION
3	BEDFORD
4	BARNETBY
5	NORMANTON
6	VERWOOD
7	GERRARDS CROSS
8	BASINGSTOKE
9	HESSLE
10	RAYLEIGH

In [64]:
```
%%sql
ALTER TABLE property_sales
ADD COLUMN towncity_key INTEGER,
ADD CONSTRAINT fk_towncity
    FOREIGN KEY (towncity_key)
    REFERENCES towncity_dimension(towncity_key);
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[64]: []

In [65]:
```
%%sql
SELECT * FROM property_sales
LIMIT 5
```

```
 * postgresql://student@/FinalProj
5 rows affected.
```

Out[65]:

price	propertytype	oldnew	duration	towncity	ppdcategorytype	date_key	count
247500	S	N	F	TOTNES	A	1429	
220000	D	N	F	YEOVIL	A	1276	
133000	T	N	F	EVESHAM	A	999	
40000	T	N	F	WEST BROMWICH	A	1383	
745000	D	N	F	HARPENDEN	A	645	

In [66]:
```
%%sql
UPDATE property_sales
SET towncity_key = (
    SELECT td.towncity_key
    FROM towncity_dimension td
    WHERE td.towncity = property_sales.towncity
);
```

```
 * postgresql://student@/FinalProj
4534439 rows affected.
```

Out[66]: []

In [67]:
```
%%sql
SELECT *
FROM property_sales
JOIN towncity_dimension ON property_sales.towncity_key = towncity_dimensi
LIMIT 12;
```

```
 * postgresql://student@/FinalProj
12 rows affected.
```

Out[67]:

price	propertytype	oldnew	duration	towncity	ppdcategorytype	date_key	co
247500	S	N	F	TOTNES	A	1429	
80000	T	N	F	LEEDS	A	243	
128000	T	N	L	SOUTHPORT	A	880	
715000	D	N	F	NEWBURY	A	858	
223500	S	N	F	CHINNOR	A	754	
240000	D	N	F	BRECON	A	1233	
182400	D	N	L	CHESTERFIELD	A	1238	
180000	F	N	L	LONDON	A	1192	
325000	D	N	F	SOUTHAMPTON	A	1655	
105000	F	N	L	EXETER	A	1238	
100000	S	N	F	MAIDSTONE	A	862	
335000	D	N	F	EXMOUTH	A	1652	

In [68]:
```
%%sql
ALTER TABLE property_sales
DROP COLUMN towncity;
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[68]: []

In [69]:
```
%%sql
SELECT * FROM property_sales
LIMIT 5
```

```
 * postgresql://student@/FinalProj
5 rows affected.
```

Out[69]:

price	propertytype	oldnew	duration	ppdcategorytype	date_key	county_key	distric
247500	S	N	F	A	1429	13	
80000	T	N	F	A	243	42	
128000	T	N	L	A	880	58	
715000	D	N	F	A	858	113	
223500	S	N	F	A	754	17	

Creating the category type dimension

In [70]:
```
%%sql
DROP TABLE IF EXISTS categorytype_dimension CASCADE;
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[70]: []

In [71]:
```
%%sql

CREATE TABLE categorytype_dimension (
    categorytype_key SERIAL PRIMARY KEY,
    categorytype VARCHAR
);
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[71]: []

In [72]:
```
%%sql
COMMENT ON COLUMN categorytype_dimension.categorytype_key IS 'The primary
COMMENT ON COLUMN categorytype_dimension.categorytype IS 'The categorytyp
```

```
 * postgresql://student@/FinalProj
Done.
Done.
```

Out[72]: []

In [73]:
```
%%sql
INSERT INTO categorytype_dimension(categorytype)
SELECT DISTINCT
    ppdcategorytype
FROM property_sales
```

```
 * postgresql://student@/FinalProj
2 rows affected.
```

Out[73]: []

In [74]:
```
%%sql
SELECT * FROM categorytype_dimension
LIMIT 10
```

```
 * postgresql://student@/FinalProj
2 rows affected.
```

Out[74]:

categorytype_key	categorytype
1	A
2	B

In [75]:
```
%%sql
ALTER TABLE property_sales
ADD COLUMN categorytype_key INTEGER,
ADD CONSTRAINT fk_categorytype
    FOREIGN KEY (categorytype_key)
    REFERENCES categorytype_dimension(categorytype_key);
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[75]: []

In [76]:
```
%%sql
SELECT * FROM property_sales
LIMIT 5
```

```
 * postgresql://student@/FinalProj
5 rows affected.
```

Out[76]:

price	propertytype	oldnew	duration	ppdcategorytype	date_key	county_key	distric
247500	S	N	F	A	1429	13	
80000	T	N	F	A	243	42	
128000	T	N	L	A	880	58	
715000	D	N	F	A	858	113	
223500	S	N	F	A	754	17	

In [77]:
```
%%sql
UPDATE property_sales
SET categorytype_key = (
    SELECT ctd.categorytype_key
    FROM categorytype_dimension ctd
    WHERE ctd.categorytype = property_sales.ppdcategorytype
);
```

```
 * postgresql://student@/FinalProj
4534439 rows affected.
```

Out[77]: []

In [78]:
```
%%sql
SELECT * FROM property_sales
LIMIT 5
```

```
 * postgresql://student@/FinalProj
5 rows affected.
```

Out[78]:

price	propertytype	oldnew	duration	ppdcategorytype	date_key	county_key	distric
247500	S	N	F	A	1429	13	
80000	T	N	F	A	243	42	
128000	T	N	L	A	880	58	
715000	D	N	F	A	858	113	
223500	S	N	F	A	754	17	

In [79]:
```
%%sql
SELECT *
FROM property_sales
JOIN categorytype_dimension ON property_sales.categorytype_key = category
LIMIT 12;
```

```
 * postgresql://student@/FinalProj
12 rows affected.
```

Out[79]:

price	propertytype	oldnew	duration	ppdcategorytype	date_key	county_key	distric
247500	S	N	F	A	1429	13	
80000	T	N	F	A	243	42	
128000	T	N	L	A	880	58	
715000	D	N	F	A	858	113	
223500	S	N	F	A	754	17	
240000	D	N	F	A	1233	49	
182400	D	N	L	A	1238	87	
180000	F	N	L	A	1192	55	
325000	D	N	F	A	1655	47	
105000	F	N	L	A	1238	13	
100000	S	N	F	A	862	6	
335000	D	N	F	A	1652	13	

In [80]:
```
%%sql
ALTER TABLE property_sales
DROP COLUMN ppdcategorytype;
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[80]: []

In [81]:
```
%%sql
SELECT * FROM property_sales
LIMIT 5
```

```
 * postgresql://student@/FinalProj
5 rows affected.
```

Out[81]:

price	propertytype	oldnew	duration	date_key	county_key	district_key	towncity_ke
247500	S	N	F	1429	13	311	25
80000	T	N	F	243	42	265	107
128000	T	N	L	880	58	41	32
715000	D	N	F	858	113	390	73
223500	S	N	F	754	17	302	54

Creating the propertytype dimension

In [82]:
```
%%sql
DROP TABLE IF EXISTS propertytype_dimension CASCADE;
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[82]: []

In [83]:
```
%%sql

CREATE TABLE propertytype_dimension (
    propertytype_key SERIAL PRIMARY KEY,
    propertytype VARCHAR
);
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[83]: []

In [84]:
```
%%sql
COMMENT ON COLUMN propertytype_dimension.postcode_key IS 'The primary key
COMMENT ON COLUMN propertytype_dimension.postcode IS 'The propertytype th
```

```
 * postgresql://student@/FinalProj
(psycopg2.errors.UndefinedColumn) column "postcode_key" of relation "pro
pertytype_dimension" does not exist

[SQL: COMMENT ON COLUMN propertytype_dimension.postcode_key IS 'The prim
ary key representing the propertytype';]
(Background on this error at: https://sqlalche.me/e/14/f405)
```

In [85]:
```
%%sql
INSERT INTO propertytype_dimension(propertytype)
SELECT DISTINCT
    propertytype
FROM property_sales
```

```
 * postgresql://student@/FinalProj
5 rows affected.
```

Out[85]: []

In [86]:
```
%%sql
SELECT * FROM propertytype_dimension
LIMIT 10
```

```
 * postgresql://student@/FinalProj
5 rows affected.
```

Out[86]:

propertytype_key	propertytype
1	S
2	O
3	T
4	F
5	D

In [87]:
```
%%sql
ALTER TABLE property_sales
ADD COLUMN propertytype_key INTEGER,
ADD CONSTRAINT fk_propertytype
    FOREIGN KEY (propertytype_key)
    REFERENCES propertytype_dimension(propertytype_key);
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[87]: []

In [88]:
```
%%sql
SELECT * FROM property_sales
LIMIT 5
```

```
 * postgresql://student@/FinalProj
5 rows affected.
```

Out[88]:

price	propertytype	oldnew	duration	date_key	county_key	district_key	towncity_ke
247500	S	N	F	1429	13	311	25
80000	T	N	F	243	42	265	107
128000	T	N	L	880	58	41	32
715000	D	N	F	858	113	390	73
223500	S	N	F	754	17	302	54

In [89]:
```
%%sql
UPDATE property_sales
SET propertytype_key = (
    SELECT ptd.propertytype_key
    FROM propertytype_dimension ptd
    WHERE ptd.propertytype = property_sales.propertytype
);
```

```
 * postgresql://student@/FinalProj
4534439 rows affected.
```

Out[89]: []

In [90]:
```
%%sql
SELECT *
FROM property_sales
JOIN propertytype_dimension ON property_sales.propertytype_key = property
LIMIT 12;
```

```
 * postgresql://student@/FinalProj
12 rows affected.
```

Out[90]:

price	propertytype	oldnew	duration	date_key	county_key	district_key	towncity_ke
247500	S	N	F	1429	13	311	25
80000	T	N	F	243	42	265	107
128000	T	N	L	880	58	41	32
715000	D	N	F	858	113	390	73
223500	S	N	F	754	17	302	54
240000	D	N	F	1233	49	307	42
182400	D	N	L	1238	87	328	84
180000	F	N	L	1192	55	129	6
325000	D	N	F	1655	47	315	73
105000	F	N	L	1238	13	277	21
100000	S	N	F	862	6	251	48
335000	D	N	F	1652	13	9	7

In [91]:
```
%%sql
ALTER TABLE property_sales
DROP COLUMN propertytype;
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[91]: []

In [92]:
```
%%sql
SELECT * FROM property_sales
LIMIT 5
```

```
 * postgresql://student@/FinalProj
5 rows affected.
```

Out[92]:

price	oldnew	duration	date_key	county_key	district_key	towncity_key	categorytyp
247500	N	F	1429	13	311	254	
80000	N	F	243	42	265	1076	
128000	N	L	880	58	41	328	
715000	N	F	858	113	390	736	
223500	N	F	754	17	302	548	

Creating the oldnew dimension

In [93]:
```
%%sql
DROP TABLE IF EXISTS oldnew_dimension CASCADE;
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[93]: []

In [94]:
```
%%sql

CREATE TABLE oldnew_dimension (
    oldnew_key SERIAL PRIMARY KEY,
    oldnew VARCHAR
);
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[94]: []

In [95]:
```
%%sql
COMMENT ON COLUMN oldnew_dimension.oldnew_key IS 'The primary key represe
COMMENT ON COLUMN oldnew_dimension.oldnew IS 'The oldnew the property is
```

```
 * postgresql://student@/FinalProj
Done.
Done.
```

Out[95]: []

In [96]:
```
%%sql
INSERT INTO oldnew_dimension(oldnew)
SELECT DISTINCT
    oldnew
FROM property_sales
```

```
 * postgresql://student@/FinalProj
2 rows affected.
```

Out[96]: []

In [97]:
```
%%sql
SELECT * FROM oldnew_dimension
LIMIT 10
```

```
 * postgresql://student@/FinalProj
2 rows affected.
```

Out[97]:

oldnew_key	oldnew
1	Y
2	N

In [98]:
```
%%sql
ALTER TABLE property_sales
ADD COLUMN oldnew_key INTEGER,
ADD CONSTRAINT fk_oldnew
    FOREIGN KEY (oldnew_key)
    REFERENCES oldnew_dimension(oldnew_key);
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[98]: []

In [99]:
```
%%sql
UPDATE property_sales
SET oldnew_key = (
    SELECT ond.oldnew_key
    FROM oldnew_dimension ond
    WHERE ond.oldnew = property_sales.oldnew
);
```

```
 * postgresql://student@/FinalProj
4534439 rows affected.
```

Out[99]: []

In [100…
```
%%sql
SELECT *
FROM property_sales
JOIN oldnew_dimension ON property_sales.oldnew_key = oldnew_dimension.old
LIMIT 12;
```

```
 * postgresql://student@/FinalProj
12 rows affected.
```

Out[100]:

price	oldnew	duration	date_key	county_key	district_key	towncity_key	categoryty
247500	N	F	1429	13	311	254	
124950	N	F	1771	12	216	504	
100000	N	L	1775	36	116	686	
83000	N	F	1305	91	281	92	
235000	N	F	1752	36	153	648	
93000	N	F	435	87	14	237	
190000	N	F	601	67	326	1118	
176000	N	L	1	55	245	60	
185000	N	L	1129	55	392	60	
51500	N	L	757	82	341	237	
240000	N	F	1416	110	134	553	
125000	N	F	1545	33	380	499	

In [101…
```
%%sql
ALTER TABLE property_sales
DROP COLUMN oldnew;
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[101]: []

In [102…
```
%%sql
SELECT * FROM property_sales
LIMIT 5
```

```
 * postgresql://student@/FinalProj
5 rows affected.
```

Out[102]:

price	duration	date_key	county_key	district_key	towncity_key	categorytype_key
247500	F	1429	13	311	254	1
124950	F	1771	12	216	504	1
100000	L	1775	36	116	686	1
83000	F	1305	91	281	92	1
235000	F	1752	36	153	648	1

Creating the duration dimension

In [103…
```
%%sql
DROP TABLE IF EXISTS duration_dimension CASCADE;
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[103]: []

In [104…
```
%%sql

CREATE TABLE duration_dimension (
    duration_key SERIAL PRIMARY KEY,
    duration VARCHAR
);
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[104]: []

In [105…
```
%%sql
COMMENT ON COLUMN duration_dimension.duration_key IS 'The primary key rep
COMMENT ON COLUMN duration_dimension.duration IS 'The duration the proper
```

```
 * postgresql://student@/FinalProj
Done.
Done.
```

Out[105]: []

In [106…
```
%%sql
INSERT INTO duration_dimension(duration)
SELECT DISTINCT
    duration
FROM property_sales
```

```
 * postgresql://student@/FinalProj
3 rows affected.
```

Out[106]: []

In [107…
```
%%sql
SELECT * FROM duration_dimension
LIMIT 10
```

```
 * postgresql://student@/FinalProj
3 rows affected.
```

Out[107]:

duration_key	duration
1	U
2	L
3	F

In [108…
```
%%sql
ALTER TABLE property_sales
ADD COLUMN duration_key INTEGER,
ADD CONSTRAINT fk_duration
    FOREIGN KEY (duration_key)
    REFERENCES duration_dimension(duration_key);
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[108]: []

In [109…
```
%%sql
SELECT * FROM property_sales
LIMIT 5
```

```
 * postgresql://student@/FinalProj
5 rows affected.
```

Out[109]:

price	duration	date_key	county_key	district_key	towncity_key	categorytype_key
247500	F	1429	13	311	254	1
124950	F	1771	12	216	504	1
100000	L	1775	36	116	686	1
83000	F	1305	91	281	92	1
235000	F	1752	36	153	648	1

In [110…
```
%%sql
UPDATE property_sales
SET duration_key = (
    SELECT drd.duration_key
    FROM duration_dimension drd
    WHERE drd.duration = property_sales.duration
);
```

```
 * postgresql://student@/FinalProj
4534439 rows affected.
```

Out[110]: []

In [111…
```
%%sql
SELECT *
FROM property_sales
JOIN duration_dimension ON property_sales.duration_key = duration_dimensi
LIMIT 12;
```

```
 * postgresql://student@/FinalProj
12 rows affected.
```

Out[111]:

price	duration	date_key	county_key	district_key	towncity_key	categorytype_key
247500	F	1429	13	311	254	1
124950	F	1771	12	216	504	1
80000	F	243	42	265	1076	1
330000	F	1300	22	243	713	1
128000	L	880	58	41	328	1
122000	F	521	36	153	813	1
715000	F	858	113	390	736	1
223500	F	754	17	302	548	1
177500	F	460	71	287	826	1
970000	F	507	17	302	418	1
240000	F	1233	49	307	427	1
185000	F	457	41	388	1160	1

In [112…
```
%%sql
ALTER TABLE property_sales
DROP COLUMN duration;
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[112]: []

In [113…
```
%%sql
SELECT * FROM property_sales
LIMIT 5
```

```
 * postgresql://student@/FinalProj
5 rows affected.
```

Out[113]:

price	date_key	county_key	district_key	towncity_key	categorytype_key	propertyty
247500	1429	13	311	254	1	
124950	1771	12	216	504	1	
80000	243	42	265	1076	1	
330000	1300	22	243	713	1	
128000	880	58	41	328	1	

Latitude and Longitude for towncity

In [114…

```
%%sql
ALTER TABLE towncity_dimension
ADD COLUMN lat NUMERIC,
ADD COLUMN lng NUMERIC;
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[114]: []

In [115…

```
%%sql
SELECT * FROM towncity_dimension ORDER BY towncity_dimension ASC LIMIT 10
```

```
 * postgresql://student@/FinalProj
10 rows affected.
```

Out[115]:

towncity_key	towncity	lat	lng
1	BRENTWOOD	None	None
2	MARAZION	None	None
3	BEDFORD	None	None
4	BARNETBY	None	None
5	NORMANTON	None	None
6	VERWOOD	None	None
7	GERRARDS CROSS	None	None
8	BASINGSTOKE	None	None
9	HESSLE	None	None
10	RAYLEIGH	None	None

In [116…
```
import psycopg2
import geocoder

conn = psycopg2.connect("dbname='FinalProj' user='student'")
c = conn.cursor()
c.execute("SELECT towncity_key, towncity FROM towncity_dimension ORDER BY
rows = c.fetchall()
for r in rows:
    towncity_key, towncity = r
    g = geocoder.arcgis('%s, towncity' % towncity)
    if g:
        c.execute("UPDATE towncity_dimension SET lat = (%s), lng = (%s) W
                  (g.lat, g.lng, towncity_key))
conn.commit()
```

There are 264 towncities where the lat and lng are none.

In [117…
```
%%sql
SELECT * FROM towncity_dimension ORDER BY towncity_key ASC LIMIT 30
```

```
 * postgresql://student@/FinalProj
30 rows affected.
```

Out[117]:

towncity_key	towncity	lat	lng
1	BRENTWOOD	40.78114000000005	-73.24647999999996
2	MARAZION	50.12351799600003	-5.473268651999945
3	BEDFORD	41.76279000000005	-83.58704999999998
4	BARNETBY	53.57589000000007	-0.4092699999999354
5	NORMANTON	62.240410000000054	10.671070000000043
6	VERWOOD	50.88186125000004	-1.879172687999926
7	GERRARDS CROSS	51.58735666600006	-0.5542401319999612
8	BASINGSTOKE	51.26775197400008	-1.0883177929999306
9	HESSLE	53.72310684200005	-0.435465044999944
10	RAYLEIGH	None	None
11	RUISLIP	None	None
12	GATESHEAD	-32.97852999999998	151.6930000000001
13	CARTERTON	None	None
14	PENZANCE	50.11843510700004	-5.539321683999958
15	CONGLETON	53.16287207500005	-2.211357343999964
16	CULLOMPTON	50.855193334000035	-3.392775588999939
17	NORTH FERRIBY	53.72222859300007	-0.5024426279999261
18	BRIGHTON	39.98670000000004	-104.81816999999995
19	LLANDUDNO JUNCTION	53.28410000000008	-3.8083299999999554
20	RIDING MILL	54.94784326200005	-1.9734227929999406
21	NEWCASTLETON	55.179305502000034	-2.8132227519999446
22	ROCHDALE	None	None
23	TAVISTOCK	53.32802000000004	-6.25816999999995
24	BLAKENEY	None	None
25	SNODLAND	51.32977000000005	0.4484400000000619
26	TENTERDEN	51.07083210500008	0.684447814000066
27	KNARESBOROUGH	54.00737607800005	-1.464599064999959
28	SUNBURY-ON-THAMES	51.40697567674323	-0.4034458209397889
29	NEWHAVEN	50.79526000000004	0.05461000000002514
30	HECKMONDWIKE	53.70713974300003	-1.6743966489999593

Latitude and Longitude for County

In [118…
```
%%sql
ALTER TABLE county_dimension
ADD COLUMN lat NUMERIC,
ADD COLUMN lng NUMERIC;
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[118]: []

In [119…
```
import psycopg2
import geocoder

conn = psycopg2.connect("dbname='FinalProj' user='student'")
c = conn.cursor()
c.execute("SELECT county_key, county FROM county_dimension ORDER BY count
rows = c.fetchall()
for r in rows:
    county_key, county = r
    g = geocoder.arcgis('%s, county' % county)
    if g:
        c.execute("UPDATE county_dimension SET lat = (%s), lng = (%s) WHE
                  (g.lat, g.lng, county_key))
conn.commit()
```

In [6]:
```
%%sql
SELECT * FROM county_dimension ORDER BY county_key ASC LIMIT 10
```

```
 * postgresql://student@/FinalProj
10 rows affected.
```

Out[6]:

county_key	county	lat	lng
1	SOUTHAMPTON	36.72029431400006	-77.10584724199998
2	BEDFORD	40.006508102000055	-78.49029097799996
3	BLAENAU GWENT	51.76677639600007	-3.1937832479999315
4	COUNTY DURHAM	54.50896000000006	-1.5702199999999493
5	PEMBROKESHIRE	51.853397037000036	-4.88911638299993
6	KENT	39.08881450300004	-75.56713029499997
7	CONWY	53.14002337000005	-3.7351635119999287
8	MERTHYR TYDFIL	51.740193368000064	-3.360921158999929
9	SOUTHEND-ON-SEA	51.533330000000035	0.7000000000000455
10	WOKINGHAM	51.41667000000007	-0.9166699999999537

Longitude and Latitudes for District

In [121…
```
%%sql
ALTER TABLE district_dimension
ADD COLUMN lat NUMERIC,
ADD COLUMN lng NUMERIC;
```

```
 * postgresql://student@/FinalProj
Done.
```

Out[121]: []

In [122…
```
import psycopg2
import geocoder

conn = psycopg2.connect("dbname='FinalProj' user='student'")
c = conn.cursor()
c.execute("SELECT district_key, district FROM district_dimension ORDER BY
rows = c.fetchall()
for r in rows:
    district_key, district = r
    g = geocoder.arcgis('%s, district' % district)
    if g:
        c.execute("UPDATE district_dimension SET lat = (%s), lng = (%s) W
                  (g.lat, g.lng, district_key))
conn.commit()
```

In [5]:
```
%%sql
SELECT * FROM district_dimension ORDER BY district_key ASC LIMIT 10
```

```
 * postgresql://student@/FinalProj
10 rows affected.
```

Out[5]:

district_key	district	lat	lng
1	BEDFORD	-32.67828999999995	26.08746000000008
2	DONCASTER	-37.77043999999995	145.13224000000002
3	OADBY AND WIGSTON	52.58567742200006	-1.1064881219999734
4	CONWY	48.85944000000006	3.8280600000000504
5	RUGBY	52.393200000000036	-1.3171199999999317
6	MERTHYR TYDFIL	51.75738056520835	-3.36591504116643
7	STAFFORDSHIRE MOORLANDS	53.10125000000005	-2.0326199999999517
8	NORTH EAST DERBYSHIRE	53.20080000000007	-1.415839999999946
9	EAST DEVON	50.75632000000007	-3.234869999999944
10	GATESHEAD	54.93333000000007	-1.6666699999999537

In [ ]:

In [ ]:

Q1)

Least Effected By County

In [5]:
```
%%sql
SELECT
    c.county,
    ROUND(pre2008.avg_price,2) As "avn_before 2008",
    ROUND(post2008.avg_price,2) As "avn_after 2008",
    ROUND(post2008.avg_price - pre2008.avg_price, 2) AS "avg_price_differ
    ROUND(((post2008.avg_price - pre2008.avg_price)*100)/post2008.avg_pri
FROM
    county_dimension c
JOIN
    (SELECT
        p.county_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year < 2008
     GROUP BY
        p.county_key) pre2008 ON c.county_key = pre2008.county_key
JOIN
    (SELECT
        p.county_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year >= 2008
     GROUP BY
        p.county_key) post2008 ON c.county_key = post2008.county_key
ORDER BY
    "Percent Change" DESC
LIMIT 10;
```

```
 * postgresql://student@/FinalProj
10 rows affected.
```

Out[5]:

county	avn_before 2008	avn_after 2008	avg_price_difference	Percent Change
BOURNEMOUTH, CHRISTCHURCH AND POOLE	270120.11	357116.38	86996.27	24.36
GREATER LONDON	334191.01	379335.48	45144.47	11.90
WINDSOR AND MAIDENHEAD	382432.32	412502.72	30070.40	7.29
HERTFORDSHIRE	283514.08	301993.43	18479.35	6.12
SURREY	352099.81	374294.54	22194.73	5.93
BUCKINGHAMSHIRE	327675.91	347722.03	20046.12	5.76
EAST SUSSEX	224074.81	237265.37	13190.56	5.56
RUTLAND	254371.88	268420.25	14048.38	5.23
BRIGHTON AND HOVE	250630.97	264159.38	13528.42	5.12
THE VALE OF GLAMORGAN	190814.17	200949.58	10135.41	5.04

Most Effected County

In [7]:
```
%%sql
SELECT
    c.county,
    ROUND(pre2008.avg_price,2) As "avn_before 2008",
    ROUND(post2008.avg_price,2) As "avn_after 2008",
    ROUND(post2008.avg_price - pre2008.avg_price, 2) AS "avg_price_differ
    ROUND(((post2008.avg_price - pre2008.avg_price)*100)/post2008.avg_pri
FROM
    county_dimension c
JOIN
    (SELECT
        p.county_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year < 2008
     GROUP BY
        p.county_key) pre2008 ON c.county_key = pre2008.county_key
JOIN
    (SELECT
        p.county_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year >= 2008
     GROUP BY
        p.county_key) post2008 ON c.county_key = post2008.county_key
ORDER BY
    "Percent Change" ASC
LIMIT 10;
```

```
 * postgresql://student@/FinalProj
10 rows affected.
```

Out[7]:

county	avn_before 2008	avn_after 2008	avg_price_difference	Percent Change
CHESHIRE EAST	291478.08	223577.47	-67900.61	-30.37
CHESHIRE WEST AND CHESTER	240822.56	196770.46	-44052.10	-22.39
COUNTY DURHAM	152417.73	127158.91	-25258.82	-19.86
CENTRAL BEDFORDSHIRE	247348.95	216129.17	-31219.78	-14.44
NEWPORT	159239.42	146140.74	-13098.68	-8.96
BEDFORD	227877.18	210863.13	-17014.05	-8.07
MERTHYR TYDFIL	108170.30	100237.55	-7932.74	-7.91
CAERPHILLY	130673.83	122197.43	-8476.40	-6.94
ISLES OF SCILLY	387331.60	362212.12	-25119.47	-6.94
BLACKPOOL	124573.97	116732.90	-7841.07	-6.72

BY TOWNCITY LEAST effected

In [13]:
```
%%sql
SELECT
    c.towncity,
    ROUND(pre2008.avg_price,2) As "avn_before 2008",
    ROUND(post2008.avg_price,2) As "avn_after 2008",
    ROUND(post2008.avg_price - pre2008.avg_price, 2) AS "avg_price_differ
    ROUND(((post2008.avg_price - pre2008.avg_price)*100)/post2008.avg_pri
FROM
    towncity_dimension c
JOIN
    (SELECT
        p.towncity_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year < 2008
     GROUP BY
        p.towncity_key) pre2008 ON c.towncity_key = pre2008.towncity_key
JOIN
    (SELECT
        p.towncity_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year >= 2008
     GROUP BY
        p.towncity_key) post2008 ON c.towncity_key = post2008.towncity_ke
ORDER BY
    "Percent Change" DESC
LIMIT 10;
```

```
 * postgresql://student@/FinalProj
10 rows affected.
```

Out[13]:

towncity	avn_before 2008	avn_after 2008	avg_price_difference	Percent Change
RAVENGLASS	260666.67	476111.11	215444.44	45.25
MARIANGLAS	260583.33	375000.00	114416.67	30.51
ARTHOG	161181.82	229762.79	68580.97	29.85
BRYNTEG	170849.90	242041.67	71191.76	29.41
LLANFYRNACH	191118.57	256414.29	65295.71	25.46
CHALFONT ST. GILES	704637.84	939643.64	235005.79	25.01
BELFORD	192355.81	254849.36	62493.55	24.52
PORT ISAAC	288620.32	380770.15	92149.83	24.20
CEMAES BAY	157684.92	205313.25	47628.33	23.20
TREFRIW	167631.58	212616.67	44985.09	21.16

Most Effected Towncity

In [8]:
```
%%sql
SELECT
    c.towncity,
    ROUND(pre2008.avg_price,2) As "avn_before 2008",
    ROUND(post2008.avg_price,2) As "avn_after 2008",
    ROUND(post2008.avg_price - pre2008.avg_price, 2) AS "avg_price_differ
    ROUND(((post2008.avg_price - pre2008.avg_price)*100)/post2008.avg_pri
FROM
    towncity_dimension c
JOIN
    (SELECT
        p.towncity_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year < 2008
     GROUP BY
        p.towncity_key) pre2008 ON c.towncity_key = pre2008.towncity_key
JOIN
    (SELECT
        p.towncity_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year >= 2008
     GROUP BY
        p.towncity_key) post2008 ON c.towncity_key = post2008.towncity_ke
ORDER BY
    "Percent Change" ASC
LIMIT 10;
```

```
 * postgresql://student@/FinalProj
10 rows affected.
```

Out[8]:

towncity	avn_before 2008	avn_after 2008	avg_price_difference	Percent Change
NEWCASTLETON	175200.00	114000.00	-61200.00	-53.68
ST NEOTS	301997.00	204754.73	-97242.27	-47.49
ST ASAPH	246444.44	185796.11	-60648.34	-32.64
ST HELENS	161945.88	122735.55	-39210.33	-31.95
HINTON ST GEORGE	454071.43	349226.19	-104845.24	-30.02
ST IVES	299303.12	233548.56	-65754.56	-28.15
PENTRAETH	210820.13	167125.00	-43695.13	-26.15
FAIRBOURNE	151724.98	122446.08	-29278.90	-23.91
OTTERY ST MARY	387681.25	314434.07	-73247.18	-23.29
LLANBEDR	198379.41	161698.33	-36681.08	-22.68

Least Effected District

In [9]:
```
%%sql
SELECT
    c.district,
    ROUND(pre2008.avg_price,2) As "avn_before 2008",
    ROUND(post2008.avg_price,2) As "avn_after 2008",
    ROUND(post2008.avg_price - pre2008.avg_price, 2) AS "avg_price_differ
    ROUND(((post2008.avg_price - pre2008.avg_price)*100)/post2008.avg_pri
FROM
    district_dimension c
JOIN
    (SELECT
        p.district_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year < 2008
     GROUP BY
        p.district_key) pre2008 ON c.district_key = pre2008.district_key
JOIN
    (SELECT
        p.district_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year >= 2008
     GROUP BY
        p.district_key) post2008 ON c.district_key = post2008.district_ke
ORDER BY
    "Percent Change" DESC
LIMIT 10;
```

```
 * postgresql://student@/FinalProj
10 rows affected.
```

Out[9]:

district	avn_before 2008	avn_after 2008	avg_price_difference	Percent Change
FOLKESTONE AND HYTHE	180000.00	240497.50	60497.50	25.16
BOURNEMOUTH, CHRISTCHURCH AND POOLE	270120.11	357116.38	86996.27	24.36
CAMDEN	525079.87	661849.81	136769.94	20.66
KENSINGTON AND CHELSEA	943775.82	1166447.62	222671.81	19.09
CITY OF WESTMINSTER	636759.09	786932.51	150173.42	19.08
HAMMERSMITH AND FULHAM	492519.77	584834.72	92314.96	15.78
MERTON	338117.44	390368.96	52251.52	13.39
BERWICK-UPON-TWEED	169598.51	193561.97	23963.45	12.38
SOUTHWARK	310416.59	351166.79	40750.20	11.60
BARNET	361920.52	408607.38	46686.86	11.43

Most Effected Districe

In [10]:
```
%%sql
SELECT
    c.district,
    ROUND(pre2008.avg_price,2) As "avn_before 2008",
    ROUND(post2008.avg_price,2) As "avn_after 2008",
    ROUND(post2008.avg_price - pre2008.avg_price, 2) AS "avg_price_differ
    ROUND(((post2008.avg_price - pre2008.avg_price)*100)/post2008.avg_pri
FROM
    district_dimension c
JOIN
    (SELECT
        p.district_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year < 2008
     GROUP BY
        p.district_key) pre2008 ON c.district_key = pre2008.district_key
JOIN
    (SELECT
        p.district_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year >= 2008
     GROUP BY
        p.district_key) post2008 ON c.district_key = post2008.district_ke
ORDER BY
    "Percent Change" ASC
LIMIT 10;
```

```
 * postgresql://student@/FinalProj
10 rows affected.
```

Out[10]:

district	avn_before 2008	avn_after 2008	avg_price_difference	Percent Change
SOMERSET WEST AND TAUNTON	309888.89	163832.50	-146056.39	-89.15
CHESHIRE EAST	291478.08	223577.47	-67900.61	-30.37
EAST SUFFOLK	256125.00	208590.00	-47535.00	-22.79
CHESHIRE WEST AND CHESTER	240425.59	196770.46	-43655.13	-22.19
WILTSHIRE	287213.97	236578.30	-50635.67	-21.40
CORNWALL	268735.75	221969.04	-46766.71	-21.07
NORTHUMBERLAND	212624.02	176359.40	-36264.62	-20.56
COUNTY DURHAM	152417.73	127158.91	-25258.82	-19.86
SHROPSHIRE	233993.68	204278.88	-29714.79	-14.55
CENTRAL BEDFORDSHIRE	247348.95	216129.17	-31219.78	-14.44

Q2)

AVG prices for year by county

In [126…

```
%%sql result <<
SELECT
    c.county,
    ROUND(y2006.avg_price,2) As "avg_price_2006",
    ROUND(y2007.avg_price,2) As "avg_price_2007",
    ROUND(y2008.avg_price,2) As "avg_price_2008",
    ROUND(y2009.avg_price,2) As "avg_price_2009",
    ROUND(y2010.avg_price,2) As "avg_price_2010"
FROM
    county_dimension c
JOIN
    (SELECT
        p.county_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year = 2006
     GROUP BY
        p.county_key) y2006 ON c.county_key = y2006.county_key
JOIN
```

```
    (SELECT
        p.county_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year = 2007
     GROUP BY
        p.county_key) y2007 ON c.county_key = y2007.county_key
JOIN
    (SELECT
        p.county_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year = 2008
     GROUP BY
        p.county_key) y2008 ON c.county_key = y2008.county_key
JOIN
    (SELECT
        p.county_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year = 2009
     GROUP BY
        p.county_key) y2009 ON c.county_key = y2009.county_key
JOIN
    (SELECT
        p.county_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year = 2010
```

```
    GROUP BY
       p.county_key) y2010 ON c.county_key = y2010.county_key

ORDER BY
    "avg_price_2008" ASC
LIMIT 10;
```

```
 * postgresql://student@/FinalProj
10 rows affected.
Returning data to local variable result
```

In [127…
```
df_result = result.DataFrame()
```

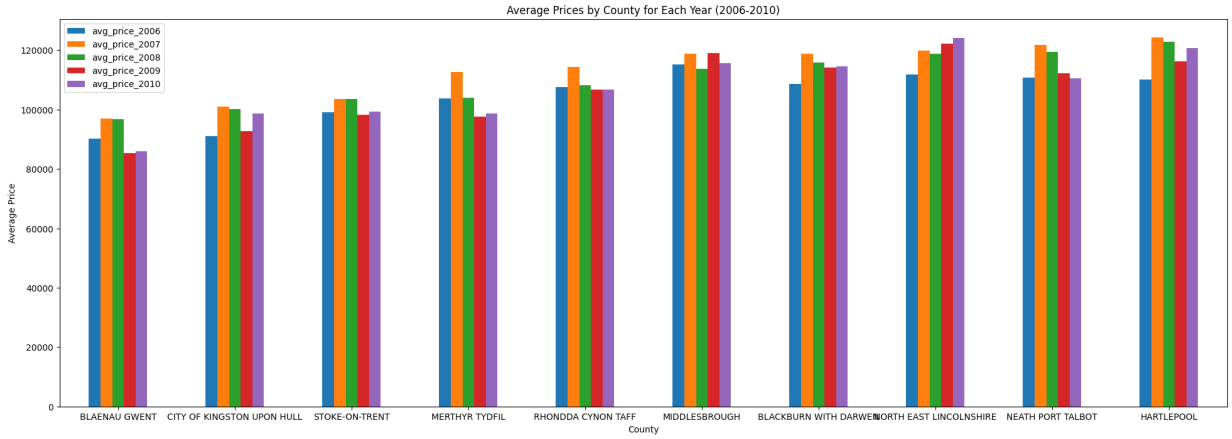
In [128…
```
df_result
```

Out[128]:

	county	avg_price_2006	avg_price_2007	avg_price_2008	avg_price_200
0	BLAENAU GWENT	90176.30	97035.11	96716.92	85370.9
1	CITY OF KINGSTON UPON HULL	91124.21	101052.59	100135.48	92766.3
2	STOKE-ON-TRENT	99167.22	103474.51	103588.46	98142.4
3	MERTHYR TYDFIL	103680.81	112705.45	103915.77	97534.7
4	RHONDDA CYNON TAFF	107597.15	114421.44	108200.99	106642.2
5	MIDDLESBROUGH	115294.62	118870.39	113789.50	119103.8
6	BLACKBURN WITH DARWEN	108639.28	118853.18	115741.04	114135.9
7	NORTH EAST LINCOLNSHIRE	111727.58	119826.61	118767.28	122261.7
8	NEATH PORT TALBOT	110803.74	121659.14	119485.97	112243.0
9	HARTLEPOOL	110174.84	124293.99	122789.59	116166.2

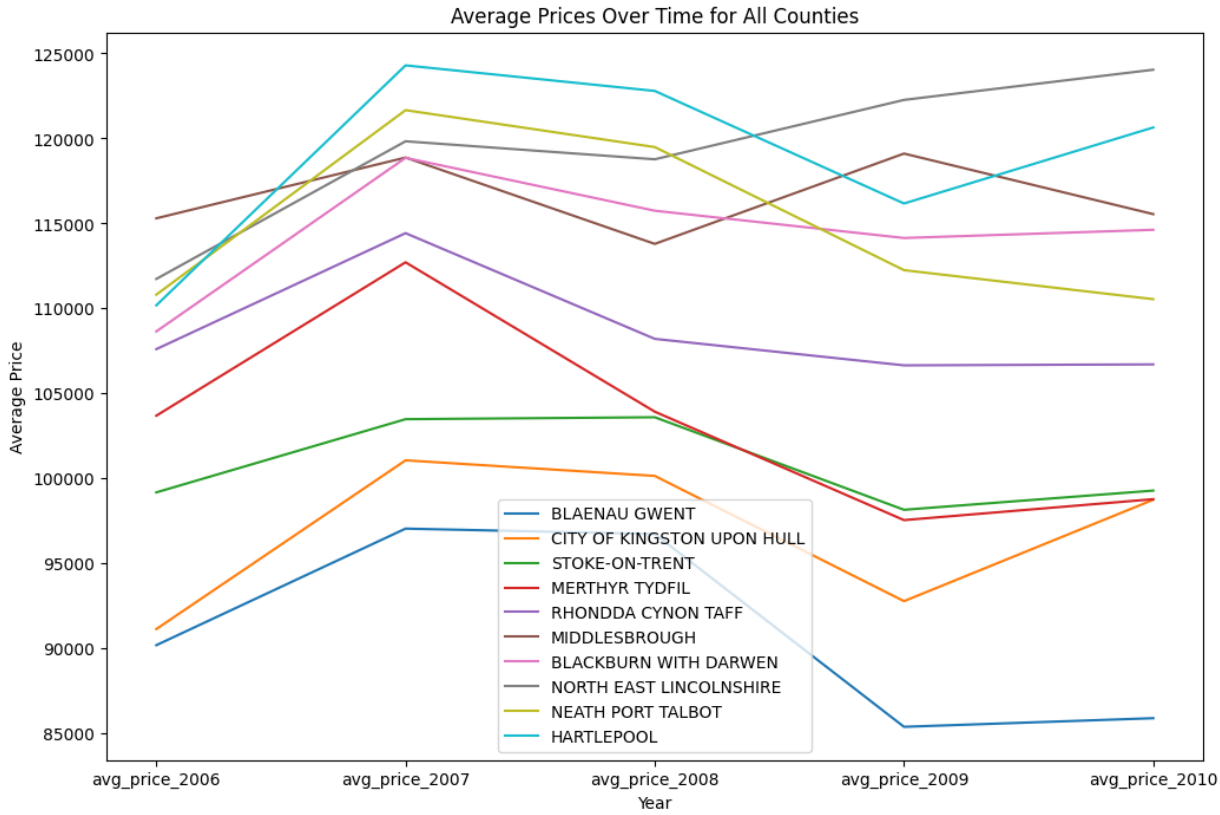
In [129…

```
import pandas as pd
import matplotlib.pyplot as plt

columns = ['avg_price_2006', 'avg_price_2007', 'avg_price_2008', 'avg_pri
df_result[columns] = df_result[columns].apply(pd.to_numeric)

index = df_result['county']

df = df_result.set_index('county')[columns]

ax = df.plot(kind='bar', rot=0, figsize=(24, 8))
ax.set_xlabel('County')
ax.set_ylabel('Average Price')
ax.set_title('Average Prices by County for Each Year (2006-2010)')

plt.show()
```



In [130…

```
import pandas as pd
import matplotlib.pyplot as plt

counties = df_result['county'].unique()
plt.figure(figsize=(12, 8))
plt.xlabel('Year')
plt.ylabel('Average Price')
plt.title('Average Prices Over Time for All Counties')

for county in counties:
    county_data = df_result[df_result['county'] == county][columns].iloc[
    plt.plot(columns, county_data, label=county)

plt.xticks(columns)
plt.legend()
plt.show()
```



AVG prices for year By City

In [131…

```
%%sql result <<
SELECT
    c.towncity,
    ROUND(y2006.avg_price,2) As "avg_price_2006",
    ROUND(y2007.avg_price,2) As "avg_price_2007",
    ROUND(y2008.avg_price,2) As "avg_price_2008",
    ROUND(y2009.avg_price,2) As "avg_price_2009",
    ROUND(y2010.avg_price,2) As "avg_price_2010"
FROM
    towncity_dimension c
JOIN
    (SELECT
        p.towncity_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year = 2006
     GROUP BY
        p.towncity_key) y2006 ON c.towncity_key = y2006.towncity_key
JOIN
    (SELECT
```

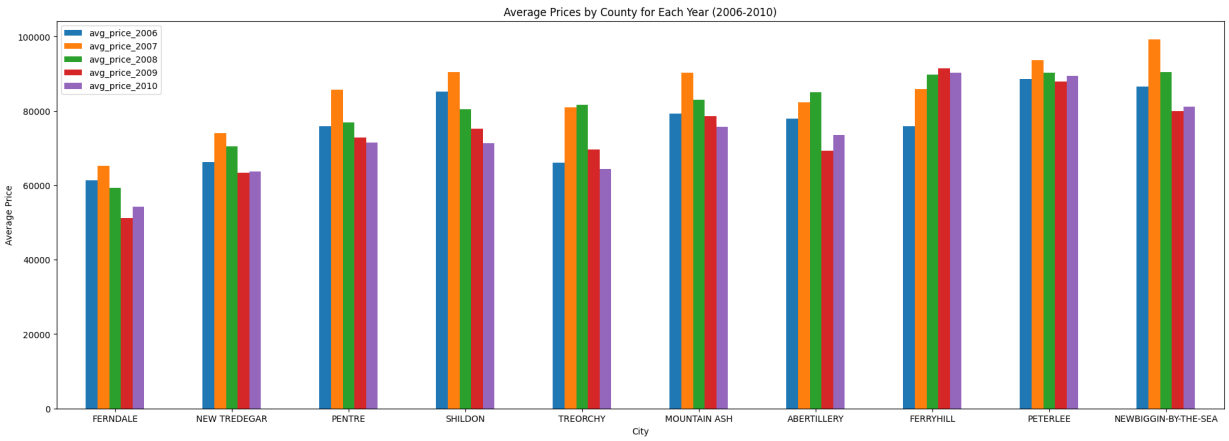
```
        p.towncity_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.year = 2007
    GROUP BY
        p.towncity_key) y2007 ON c.towncity_key = y2007.towncity_key
JOIN
    (SELECT
        p.towncity_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.year = 2008
    GROUP BY
        p.towncity_key) y2008 ON c.towncity_key = y2008.towncity_key
JOIN
    (SELECT
        p.towncity_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.year = 2009
    GROUP BY
        p.towncity_key) y2009 ON c.towncity_key = y2009.towncity_key
JOIN
    (SELECT
        p.towncity_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.year = 2010
    GROUP BY
```

```
        p.towncity_key) y2010 ON c.towncity_key = y2010.towncity_key

ORDER BY
    "avg_price_2008" ASC
LIMIT 10;
```

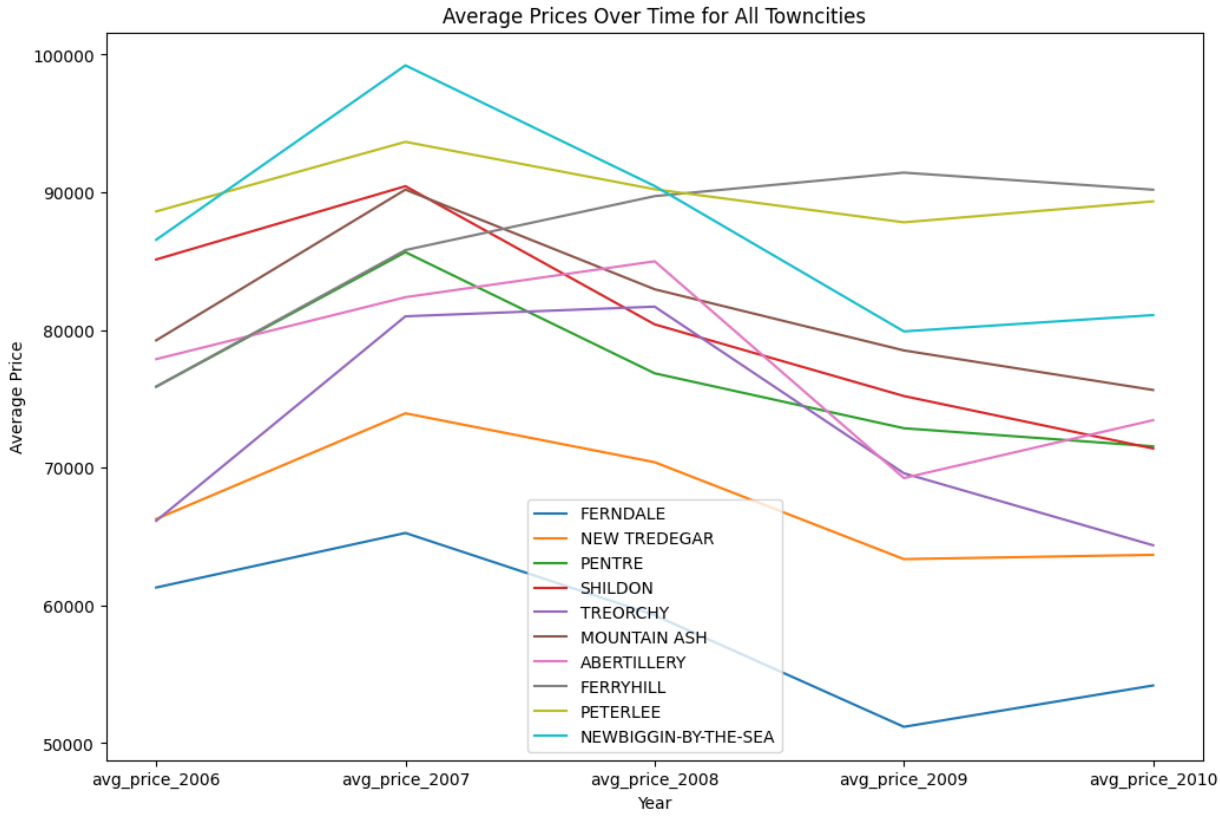
```
 * postgresql://student@/FinalProj
10 rows affected.
Returning data to local variable result
```

In [132…
```
df_result1 = result.DataFrame()
```

In [133…
```
import pandas as pd
import matplotlib.pyplot as plt

columns = ['avg_price_2006', 'avg_price_2007', 'avg_price_2008', 'avg_pri
df_result1[columns] = df_result1[columns].apply(pd.to_numeric)

index = df_result1['towncity']

df = df_result1.set_index('towncity')[columns]

ax = df.plot(kind='bar', rot=0, figsize=(24, 8))
ax.set_xlabel('City')
ax.set_ylabel('Average Price')
ax.set_title('Average Prices by County for Each Year (2006-2010)')

plt.show()
```



In [134…
```
import pandas as pd
import matplotlib.pyplot as plt

towncity = df_result1['towncity'].unique()
plt.figure(figsize=(12, 8))
plt.xlabel('Year')
plt.ylabel('Average Price')
plt.title('Average Prices Over Time for All Towncities')

for towncity in towncity:
    towncity_data = df_result1[df_result1['towncity'] == towncity][column
    plt.plot(columns, towncity_data, label=towncity)

plt.xticks(columns)
plt.legend()
plt.show()
```



AVG prices for year By District

In [135…
```
%%sql result <<
SELECT
    c.district,
    ROUND(y2006.avg_price,2) As "avg_price_2006",
    ROUND(y2007.avg_price,2) As "avg_price_2007",
    ROUND(y2008.avg_price,2) As "avg_price_2008",
    ROUND(y2009.avg_price,2) As "avg_price_2009",
    ROUND(y2010.avg_price,2) As "avg_price_2010"
FROM
```

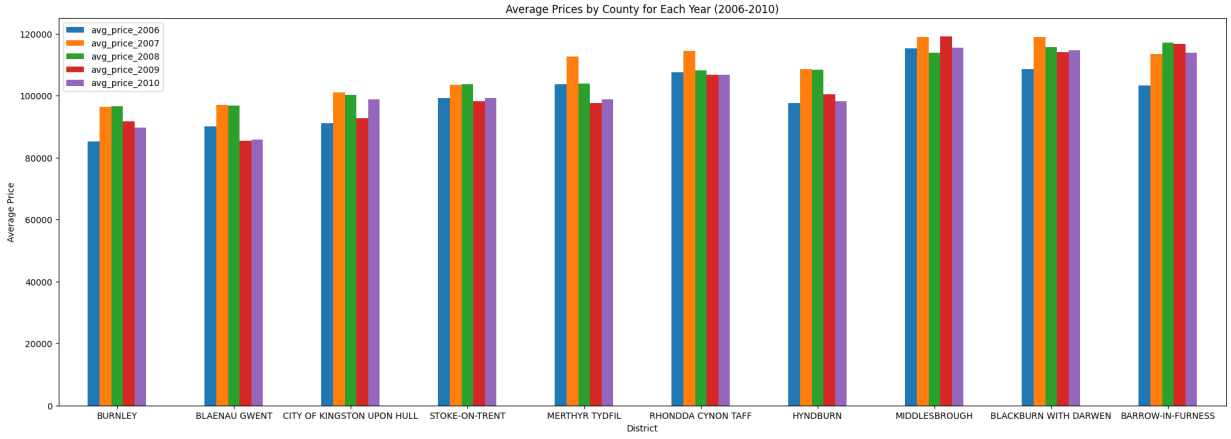
```
    district_dimension c
JOIN
    (SELECT
        p.district_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year = 2006
     GROUP BY
        p.district_key) y2006 ON c.district_key = y2006.district_key
JOIN
    (SELECT
        p.district_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year = 2007
     GROUP BY
        p.district_key) y2007 ON c.district_key = y2007.district_key
JOIN
    (SELECT
        p.district_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year = 2008
     GROUP BY
        p.district_key) y2008 ON c.district_key = y2008.district_key
JOIN
    (SELECT
        p.district_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
```

```
     WHERE
        d.year = 2009
     GROUP BY
        p.district_key) y2009 ON c.district_key = y2009.district_key
JOIN
    (SELECT
        p.district_key,
        COUNT(*) AS individual_count,
        SUM(p.price) AS total_sum,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year = 2010
     GROUP BY
        p.district_key) y2010 ON c.district_key = y2010.district_key

ORDER BY
    "avg_price_2008" ASC
LIMIT 10;
```

```
 * postgresql://student@/FinalProj
10 rows affected.
Returning data to local variable result
```

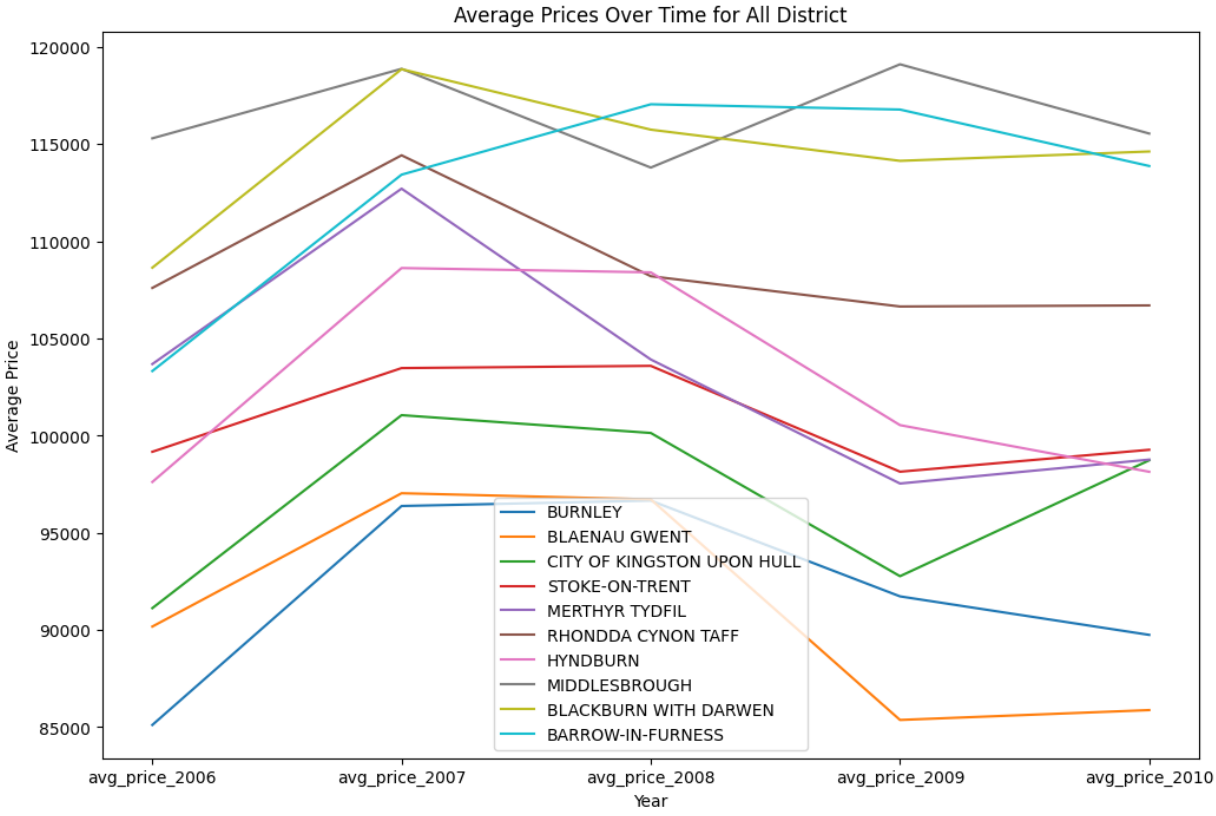
In [136…
```
df_result2 = result.DataFrame()
```

In [137…
```
import pandas as pd
import matplotlib.pyplot as plt

columns = ['avg_price_2006', 'avg_price_2007', 'avg_price_2008', 'avg_pri
df_result2[columns] = df_result2[columns].apply(pd.to_numeric)

index = df_result2['district']

df = df_result2.set_index('district')[columns]

ax = df.plot(kind='bar', rot=0, figsize=(24, 8))
ax.set_xlabel('District')
ax.set_ylabel('Average Price')
ax.set_title('Average Prices by County for Each Year (2006-2010)')

plt.show()
```



In [138…

```
import pandas as pd
import matplotlib.pyplot as plt

district = df_result2['district'].unique()
plt.figure(figsize=(12, 8))
plt.xlabel('Year')
plt.ylabel('Average Price')
plt.title('Average Prices Over Time for All District')

for district in district:
    district_data = df_result2[df_result2['district'] == district][column
    plt.plot(columns, district_data, label=district)

plt.xticks(columns)
plt.legend()
plt.show()
```



Q2)

From the above observations i see the prices are down in 2009

Now working on the particular 2009 year for top ten counties and also the avg price of old and new prop by counties in year 2009

By County

In [6]:
```
%%sql
SELECT
    c.county,
    ROUND(y2009.avg_price, 2) As "avg_price_2009",
    ROUND(m1.avg_price, 2) As "Jan",
    ROUND(m2.avg_price, 2) As "Feb",
    ROUND(m3.avg_price, 2) As "Mar",
    ROUND(m4.avg_price, 2) As "April",
    ROUND(m5.avg_price, 2) As "May",
    ROUND(m6.avg_price, 2) As "June",
    ROUND(m7.avg_price, 2) As "July",
    ROUND(m8.avg_price, 2) As "Aug",
    ROUND(m9.avg_price, 2) As "Sep",
    ROUND(m10.avg_price, 2) As "Oct",
    ROUND(m11.avg_price, 2) As "Nov",
    ROUND(m12.avg_price, 2) As "Dec"
FROM
    county_dimension c
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.year = 2009
     GROUP BY
        p.county_key) y2009 ON c.county_key = y2009.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
     FROM
```

```
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 1 and d.year = 2009
    GROUP BY
        p.county_key) m1 ON c.county_key = m1.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 2 and d.year = 2009
    GROUP BY
        p.county_key) m2 ON c.county_key = m2.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 3 and d.year = 2009
    GROUP BY
        p.county_key) m3 ON c.county_key = m3.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 4 and d.year = 2009
    GROUP BY
        p.county_key) m4 ON c.county_key = m4.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 5 and d.year = 2009
```

```
     GROUP BY
        p.county_key) m5 ON c.county_key = m5.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 6 and d.year = 2009
     GROUP BY
        p.county_key) m6 ON c.county_key = m6.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 7 and d.year = 2009
     GROUP BY
        p.county_key) m7 ON c.county_key = m7.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 8 and d.year = 2009
     GROUP BY
        p.county_key) m8 ON c.county_key = m8.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 9 and d.year = 2009
     GROUP BY
        p.county_key) m9 ON c.county_key = m9.county_key
JOIN
    (SELECT
        p.county_key,
```

```
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 10 and d.year = 2009
    GROUP BY
        p.county_key) m10 ON c.county_key = m10.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 11 and d.year = 2009
    GROUP BY
        p.county_key) m11 ON c.county_key = m11.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 12 and d.year = 2009
    GROUP BY
        p.county_key) m12 ON c.county_key = m12.county_key
ORDER BY
    "avg_price_2009" ASC
LIMIT 10;
```

```
 * postgresql://student@/FinalProj
10 rows affected.
```

Out[6]:

county	avg_price_2009	Jan	Feb	Mar	April	May
BLAENAU GWENT	85370.92	96000.00	79972.04	76934.62	76956.72	75422.82
CITY OF KINGSTON UPON HULL	92766.31	84234.08	82144.08	87272.68	77555.97	88960.10
MERTHYR TYDFIL	97534.78	94764.75	93825.00	105696.04	121737.04	86104.33
STOKE-ON-TRENT	98142.40	96608.48	84849.92	88870.68	100842.14	94801.13
RHONDDA CYNON TAFF	106642.29	101543.39	102419.64	104826.00	108063.44	102153.01
BLACKPOOL	111029.81	99853.19	117266.45	107160.77	111863.65	111163.35
NEATH PORT TALBOT	112243.00	108080.47	91679.62	112850.88	116006.08	100984.36
BLACKBURN WITH DARWEN	114135.93	109443.64	112694.37	101339.80	100315.36	113261.85
HARTLEPOOL	116166.24	98057.67	84626.14	106083.80	96647.46	113115.67
CAERPHILLY	118043.93	97944.80	116750.78	115897.58	113509.55	120758.81

In [7]:

```
%%sql result <<
SELECT
    c.county,
    ROUND(y2009.avg_price, 2) As "avg_price_2009",
    ROUND(m1.avg_price, 2) As "Jan",
    ROUND(m2.avg_price, 2) As "Feb",
    ROUND(m3.avg_price, 2) As "Mar",
    ROUND(m4.avg_price, 2) As "April",
    ROUND(m5.avg_price, 2) As "May",
    ROUND(m6.avg_price, 2) As "June",
    ROUND(m7.avg_price, 2) As "July",
    ROUND(m8.avg_price, 2) As "Aug",
    ROUND(m9.avg_price, 2) As "Sep",
    ROUND(m10.avg_price, 2) As "Oct",
    ROUND(m11.avg_price, 2) As "Nov",
    ROUND(m12.avg_price, 2) As "Dec"
FROM
    county_dimension c
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
```

```
        d.year = 2009
    GROUP BY
        p.county_key) y2009 ON c.county_key = y2009.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 1 and d.year = 2009
    GROUP BY
        p.county_key) m1 ON c.county_key = m1.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 2 and d.year = 2009
    GROUP BY
        p.county_key) m2 ON c.county_key = m2.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 3 and d.year = 2009
    GROUP BY
        p.county_key) m3 ON c.county_key = m3.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 4 and d.year = 2009
    GROUP BY
        p.county_key) m4 ON c.county_key = m4.county_key
JOIN
    (SELECT
```

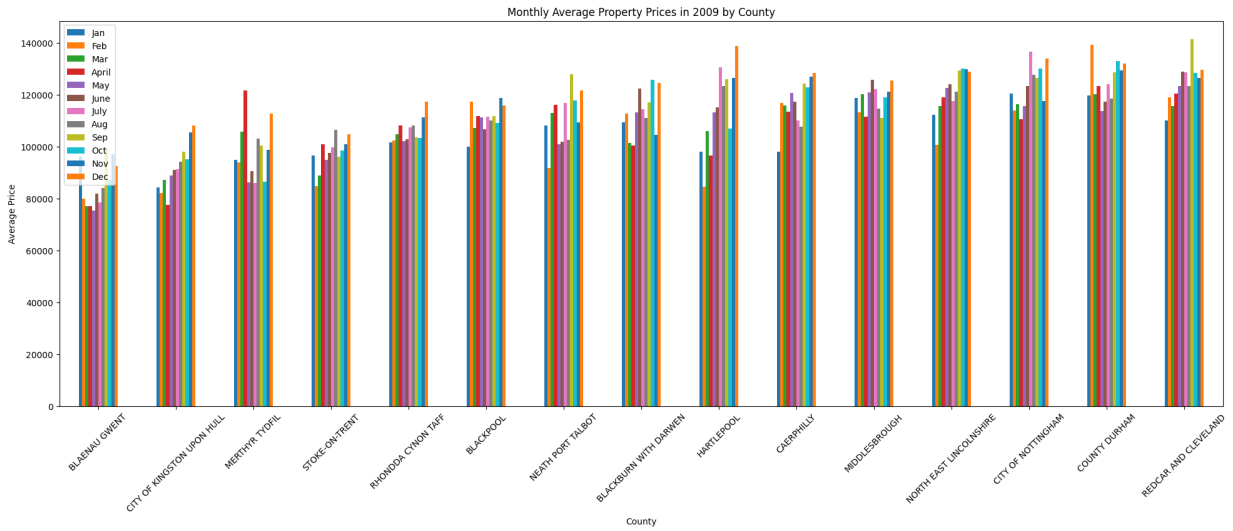
```
        p.county_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 5 and d.year = 2009
    GROUP BY
        p.county_key) m5 ON c.county_key = m5.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 6 and d.year = 2009
    GROUP BY
        p.county_key) m6 ON c.county_key = m6.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 7 and d.year = 2009
    GROUP BY
        p.county_key) m7 ON c.county_key = m7.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 8 and d.year = 2009
    GROUP BY
        p.county_key) m8 ON c.county_key = m8.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
```

```
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 9 and d.year = 2009
     GROUP BY
        p.county_key) m9 ON c.county_key = m9.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 10 and d.year = 2009
     GROUP BY
        p.county_key) m10 ON c.county_key = m10.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 11 and d.year = 2009
     GROUP BY
        p.county_key) m11 ON c.county_key = m11.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 12 and d.year = 2009
     GROUP BY
        p.county_key) m12 ON c.county_key = m12.county_key
ORDER BY
    "avg_price_2009" ASC
LIMIT 15;
```

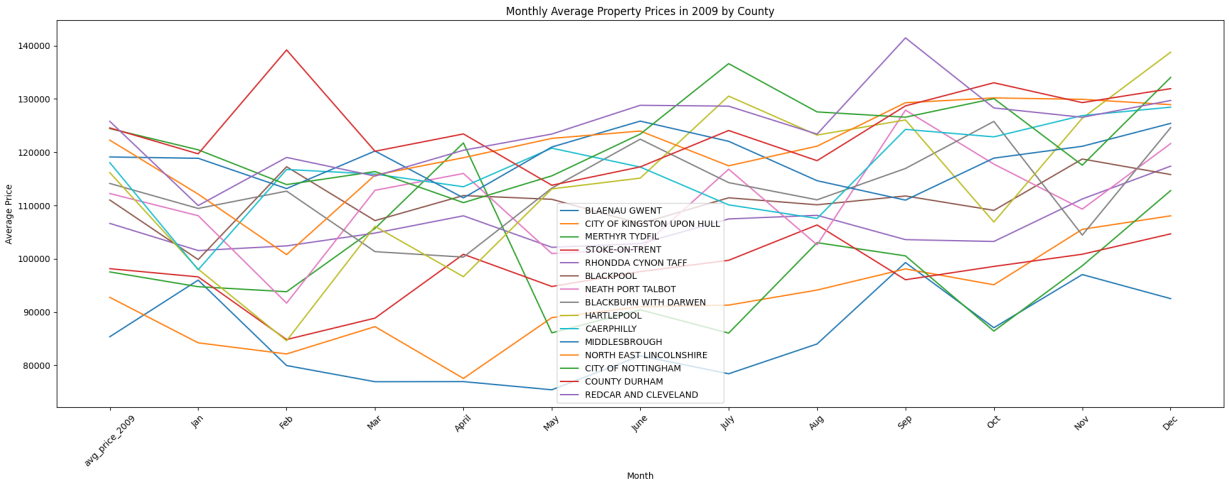
```
 * postgresql://student@/FinalProj
15 rows affected.
Returning data to local variable result
```

In [8]:
```
df_result3 = result.DataFrame()
```

In [9]:
```
import pandas as pd
import matplotlib.pyplot as plt

# List of columns as per your SQL query
columns = ['Jan', 'Feb', 'Mar', 'April', 'May', 'June',
           'July', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

# Assuming df_result is the DataFrame containing the results of your SQL
df_result3[columns] = df_result3[columns].apply(pd.to_numeric)

# Setting the index to the county
df = df_result3.set_index('county')

# Plotting the data
ax = df[columns].plot(kind='bar', rot=45, figsize=(24, 8))
ax.set_xlabel('County')
ax.set_ylabel('Average Price')
ax.set_title('Monthly Average Property Prices in 2009 by County')

plt.show()
```



In [10]:
```
import pandas as pd
import matplotlib.pyplot as plt

columns = ['avg_price_2009', 'Jan', 'Feb', 'Mar', 'April', 'May', 'June',
           'July', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

df_result3[columns] = df_result3[columns].apply(pd.to_numeric)

df = df_result3.set_index('county')
plt.figure(figsize=(24, 8))
for index, row in df.iterrows():
    plt.plot(columns, row, label=index)

plt.xlabel('Month')
plt.ylabel('Average Price')
plt.title('Monthly Average Property Prices in 2009 by County')
plt.xticks(columns, rotation=45)
plt.legend()
plt.show()
```



For a particular county

In [145…

```
%%sql
SELECT
    c.county,
    new_properties.avg_price_new,
    old_properties.avg_price_old
FROM
    county_dimension c
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price_new
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     JOIN
        oldnew_dimension ond ON p.oldnew_key = ond.oldnew_key
     WHERE
        d.year = 2009 AND ond.oldnew = 'Y'
     GROUP BY
        p.county_key) new_properties ON c.county_key = new_properties.cou
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price_old
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     JOIN
        oldnew_dimension ond ON p.oldnew_key = ond.oldnew_key
     WHERE
        d.year = 2009 AND ond.oldnew = 'N'
     GROUP BY
        p.county_key) old_properties ON c.county_key = old_properties.cou
ORDER BY
    "avg_price_new" ASC
LIMIT 15;
```

```
 * postgresql://student@/FinalProj
15 rows affected.
```

Out[145]:

county	avg_price_new	avg_price_old
BLACKPOOL	111473.066666666667	111006.451862262825
STOKE-ON-TRENT	112661.794642857143	96541.050221565731
MERTHYR TYDFIL	114848.155172413793	92345.240310077519
BLACKBURN WITH DARWEN	122179.300000000000	113516.737490377213
NEWPORT	124237.391705069124	143541.981624758221
LUTON	124444.444444444444	153818.994495412844
DURHAM	127355.700000000000	104548.064516129032
BLAENAU GWENT	129675.966666666667	82329.382151029748
CITY OF NOTTINGHAM	131137.445833333333	123757.697478991597
MIDDLESBROUGH	133374.837209302326	117340.456896551724
CITY OF KINGSTON UPON HULL	133866.537037037037	89634.490592662277
HARTLEPOOL	138257.764285714286	111563.842261904762
WREXHAM	138682.933797909408	151641.069743589744
REDCAR AND CLEVELAND	140371.402173913043	124618.811188811189
CAERPHILLY	141132.846153846154	114568.176178660050

BY TOWNCITY

In [11]:

```
%%sql
SELECT
    c.towncity,
    ROUND(y2009.avg_price, 2) As "avg_price_2009",
    ROUND(m1.avg_price, 2) As "Jan",
    ROUND(m2.avg_price, 2) As "Feb",
    ROUND(m3.avg_price, 2) As "Mar",
    ROUND(m4.avg_price, 2) As "April",
    ROUND(m5.avg_price, 2) As "May",
    ROUND(m6.avg_price, 2) As "June",
    ROUND(m7.avg_price, 2) As "July",
    ROUND(m8.avg_price, 2) As "Aug",
    ROUND(m9.avg_price, 2) As "Sep",
    ROUND(m10.avg_price, 2) As "Oct",
    ROUND(m11.avg_price, 2) As "Nov",
    ROUND(m12.avg_price, 2) As "Dec"
FROM
    towncity_dimension c
JOIN
    (SELECT
        p.towncity_key,
        AVG(p.price) AS avg_price
     FROM
```

```
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.year = 2009
    GROUP BY
        p.towncity_key) y2009 ON c.towncity_key = y2009.towncity_key
JOIN
    (SELECT
        p.towncity_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 1 and d.year = 2009
    GROUP BY
        p.towncity_key) m1 ON c.towncity_key = m1.towncity_key
JOIN
    (SELECT
        p.towncity_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 2 and d.year = 2009
    GROUP BY
        p.towncity_key) m2 ON c.towncity_key = m2.towncity_key
JOIN
    (SELECT
        p.towncity_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 3 and d.year = 2009
    GROUP BY
        p.towncity_key) m3 ON c.towncity_key = m3.towncity_key
JOIN
    (SELECT
        p.towncity_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 4 and d.year = 2009
```

```
     GROUP BY
        p.towncity_key) m4 ON c.towncity_key = m4.towncity_key
JOIN
    (SELECT
        p.towncity_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 5 and d.year = 2009
     GROUP BY
        p.towncity_key) m5 ON c.towncity_key = m5.towncity_key
JOIN
    (SELECT
        p.towncity_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 6 and d.year = 2009
     GROUP BY
        p.towncity_key) m6 ON c.towncity_key = m6.towncity_key
JOIN
    (SELECT
        p.towncity_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 7 and d.year = 2009
     GROUP BY
        p.towncity_key) m7 ON c.towncity_key = m7.towncity_key
JOIN
    (SELECT
        p.towncity_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 8 and d.year = 2009
     GROUP BY
        p.towncity_key) m8 ON c.towncity_key = m8.towncity_key
JOIN
    (SELECT
        p.towncity_key,
```

```
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 9 and d.year = 2009
    GROUP BY
        p.towncity_key) m9 ON c.towncity_key = m9.towncity_key
JOIN
    (SELECT
        p.towncity_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 10 and d.year = 2009
    GROUP BY
        p.towncity_key) m10 ON c.towncity_key = m10.towncity_key
JOIN
    (SELECT
        p.towncity_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 11 and d.year = 2009
    GROUP BY
        p.towncity_key) m11 ON c.towncity_key = m11.towncity_key
JOIN
    (SELECT
        p.towncity_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 12 and d.year = 2009
    GROUP BY
        p.towncity_key) m12 ON c.towncity_key = m12.towncity_key
ORDER BY
    "avg_price_2009" ASC
LIMIT 10;
```

```
 * postgresql://student@/FinalProj
10 rows affected.
```

Out[11]:

towncity	avg_price_2009	Jan	Feb	Mar	April	May	
FERNDALE	51173.48	48000.00	50000.00	35125.00	56809.29	39833.33	51
ABERTILLERY	69243.74	133200.00	57500.00	60438.24	58590.91	67500.00	7
TREORCHY	69592.50	67142.86	39000.00	78322.22	63000.00	70787.50	89
BARGOED	71740.07	81089.80	55333.33	72827.78	70500.00	79291.00	74
PENTRE	72861.92	108750.00	61940.33	64800.00	110666.67	61500.00	58
SHILDON	75201.45	85190.00	69714.29	72800.00	101700.00	67666.67	65
TONYPANDY	75433.86	83748.75	69272.73	75977.27	94051.64	78927.86	87
MOUNTAIN ASH	78514.20	58700.00	69998.00	105035.00	91055.56	53818.18	75
NEWBIGGIN-BY-THE-SEA	79895.75	69250.00	52625.00	83817.27	149997.50	64237.50	86
TREDEGAR	87558.92	95250.00	62000.00	89400.00	81894.50	62627.64	81

Month avg prices from last quater of 2008 to first quarter of 2009 by county

In [14]:

```
%%sql
SELECT
    c.county,
    ROUND(m1.avg_price, 2) As "Jan_2009",
    ROUND(m2.avg_price, 2) As "Feb_2009",
    ROUND(m3.avg_price, 2) As "Mar_2009",
    ROUND(m4.avg_price, 2) As "April_2009",
    ROUND(m9.avg_price, 2) As "Sep_2008",
    ROUND(m10.avg_price, 2) As "Oct_2008",
    ROUND(m11.avg_price, 2) As "Nov_2008",
    ROUND(m12.avg_price, 2) As "Dec_2008"
FROM
    county_dimension c
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 1 and d.year = 2009
     GROUP BY
        p.county_key) m1 ON c.county_key = m1.county_key
JOIN
    (SELECT
        p.county_key,
```

```
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 2 and d.year = 2009
    GROUP BY
        p.county_key) m2 ON c.county_key = m2.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 3 and d.year = 2009
    GROUP BY
        p.county_key) m3 ON c.county_key = m3.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 4 and d.year = 2009
    GROUP BY
        p.county_key) m4 ON c.county_key = m4.county_key

JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
        date_dimension d ON p.date_key = d.date_key
    WHERE
        d.month = 9 and d.year = 2008
    GROUP BY
        p.county_key) m9 ON c.county_key = m9.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
    FROM
        property_sales p
    JOIN
```

```
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 10 and d.year = 2008
     GROUP BY
        p.county_key) m10 ON c.county_key = m10.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 11 and d.year = 2008
     GROUP BY
        p.county_key) m11 ON c.county_key = m11.county_key
JOIN
    (SELECT
        p.county_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 12 and d.year = 2008
     GROUP BY
        p.county_key) m12 ON c.county_key = m12.county_key
ORDER BY
    "Jan_2009" ASC
LIMIT 15;
```

```
 * postgresql://student@/FinalProj
15 rows affected.
```

Out[14]:

county	Jan_2009	Feb_2009	Mar_2009	April_2009	Sep_2008	Oct_2008	I
CITY OF KINGSTON UPON HULL	84234.08	82144.08	87272.68	77555.97	95183.78	97815.69	
MERTHYR TYDFIL	94764.75	93825.00	105696.04	121737.04	104069.61	92028.84	
BLAENAU GWENT	96000.00	79972.04	76934.62	76956.72	85310.50	87671.60	
STOKE-ON-TRENT	96608.48	84849.92	88870.68	100842.14	106667.81	98053.52	
CAERPHILLY	97944.80	116750.78	115897.58	113509.55	117431.34	119010.86	
HARTLEPOOL	98057.67	84626.14	106083.80	96647.46	114374.05	140821.86	
BLACKPOOL	99853.19	117266.45	107160.77	111863.65	126044.56	110138.81	
RHONDDA CYNON TAFF	101543.39	102419.64	104826.00	108063.44	101608.32	102812.65	
TORFAEN	107190.91	157958.16	140506.89	114586.94	135763.62	122163.71	
NEATH PORT TALBOT	108080.47	91679.62	112850.88	116006.08	121476.61	106510.92	
BLACKBURN WITH DARWEN	109443.64	112694.37	101339.80	100315.36	112405.11	119792.21	
REDCAR AND CLEVELAND	109986.27	119018.93	115518.83	120362.58	114609.87	132385.70	
NORTH EAST LINCOLNSHIRE	112130.68	100784.86	115704.47	118964.14	123238.52	109813.92	
DARLINGTON	115691.18	123702.83	144352.05	124099.32	133937.24	139406.09	
MIDDLESBROUGH	118854.73	113175.19	120214.07	111428.06	119886.43	103484.29	

Month avg prices from last quater of 2008 to first quarter of 2009 by Towncity

In [15]:

```
%%sql
SELECT
    c.towncity,
    ROUND(m1.avg_price, 2) As "Jan_2009",
    ROUND(m2.avg_price, 2) As "Feb_2009",
    ROUND(m3.avg_price, 2) As "Mar_2009",
    ROUND(m4.avg_price, 2) As "April_2009",
    ROUND(m9.avg_price, 2) As "Sep_2008",
    ROUND(m10.avg_price, 2) As "Oct_2008",
    ROUND(m11.avg_price, 2) As "Nov_2008",
    ROUND(m12.avg_price, 2) As "Dec_2008"
FROM
    towncity_dimension c
JOIN
    (SELECT
```

```
        p.towncity_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 1 and d.year = 2009
     GROUP BY
        p.towncity_key) m1 ON c.towncity_key = m1.towncity_key
JOIN
    (SELECT
        p.towncity_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 2 and d.year = 2009
     GROUP BY
        p.towncity_key) m2 ON c.towncity_key = m2.towncity_key
JOIN
    (SELECT
        p.towncity_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 3 and d.year = 2009
     GROUP BY
        p.towncity_key) m3 ON c.towncity_key = m3.towncity_key
JOIN
    (SELECT
        p.towncity_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 4 and d.year = 2009
     GROUP BY
        p.towncity_key) m4 ON c.towncity_key = m4.towncity_key

JOIN
    (SELECT
        p.towncity_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
```

```
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 9 and d.year = 2008
     GROUP BY
        p.towncity_key) m9 ON c.towncity_key = m9.towncity_key
JOIN
    (SELECT
        p.towncity_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 10 and d.year = 2008
     GROUP BY
        p.towncity_key) m10 ON c.towncity_key = m10.towncity_key
JOIN
    (SELECT
        p.towncity_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 11 and d.year = 2008
     GROUP BY
        p.towncity_key) m11 ON c.towncity_key = m11.towncity_key
JOIN
    (SELECT
        p.towncity_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 12 and d.year = 2008
     GROUP BY
        p.towncity_key) m12 ON c.towncity_key = m12.towncity_key
ORDER BY
    "Jan_2009" ASC
LIMIT 15;
```

```
 * postgresql://student@/FinalProj
15 rows affected.
```

Out[15]:

towncity	Jan_2009	Feb_2009	Mar_2009	April_2009	Sep_2008	Oct_2008	Nov
FERNDALE	48000.00	50000.00	35125.00	56809.29	54437.50	38166.67	40
MOUNTAIN ASH	58700.00	69998.00	105035.00	91055.56	68004.12	94505.57	82
TREORCHY	67142.86	39000.00	78322.22	63000.00	92358.30	64222.22	49
PETERLEE	67889.29	82621.43	73274.38	82325.00	95440.38	90687.93	86
NEWBIGGIN-BY-THE-SEA	69250.00	52625.00	83817.27	149997.50	117862.50	90333.33	79
CRADLEY HEATH	75500.00	135857.14	128000.00	95068.75	122713.64	108857.14	101
EBBW VALE	75750.00	94327.81	89777.78	92732.14	91251.56	88320.22	99
BLACKWOOD	75818.18	128912.78	125394.72	90797.50	128499.71	122293.75	122
PORTHMADOG	76375.00	168000.00	128333.33	198000.00	186333.33	147214.29	182
FERRYHILL	77800.00	90099.38	73654.55	80412.50	191359.94	71353.13	97
MAESTEG	80000.00	87214.29	70500.00	87229.41	74236.64	104904.17	78
BOOTLE	80278.57	96322.73	87345.77	85663.23	96716.72	91388.33	97
BARGOED	81089.80	55333.33	72827.78	70500.00	81807.14	98540.00	116
MARYPORT	83494.44	151610.56	105540.91	112207.14	118171.92	98491.67	103
TONYPANDY	83748.75	69272.73	75977.27	94051.64	97465.31	77896.25	88

Month avg prices from last quater of 2008 to first quarter of 2009 By District

In [17]:

```
%%sql
SELECT
    c.district,
    ROUND(m1.avg_price, 2) As "Jan_2009",
    ROUND(m2.avg_price, 2) As "Feb_2009",
    ROUND(m3.avg_price, 2) As "Mar_2009",
    ROUND(m4.avg_price, 2) As "April_2009",
    ROUND(m9.avg_price, 2) As "Sep_2008",
    ROUND(m10.avg_price, 2) As "Oct_2008",
    ROUND(m11.avg_price, 2) As "Nov_2008",
    ROUND(m12.avg_price, 2) As "Dec_2008"
FROM
    district_dimension c
JOIN
    (SELECT
        p.district_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
```

```
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 1 and d.year = 2009
     GROUP BY
        p.district_key) m1 ON c.district_key = m1.district_key
JOIN
    (SELECT
        p.district_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 2 and d.year = 2009
     GROUP BY
        p.district_key) m2 ON c.district_key = m2.district_key
JOIN
    (SELECT
        p.district_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 3 and d.year = 2009
     GROUP BY
        p.district_key) m3 ON c.district_key = m3.district_key
JOIN
    (SELECT
        p.district_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 4 and d.year = 2009
     GROUP BY
        p.district_key) m4 ON c.district_key = m4.district_key

JOIN
    (SELECT
        p.district_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 9 and d.year = 2008
```

```
     GROUP BY
        p.district_key) m9 ON c.district_key = m9.district_key
JOIN
    (SELECT
        p.district_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 10 and d.year = 2008
     GROUP BY
        p.district_key) m10 ON c.district_key = m10.district_key
JOIN
    (SELECT
        p.district_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 11 and d.year = 2008
     GROUP BY
        p.district_key) m11 ON c.district_key = m11.district_key
JOIN
    (SELECT
        p.district_key,
        AVG(p.price) AS avg_price
     FROM
        property_sales p
     JOIN
        date_dimension d ON p.date_key = d.date_key
     WHERE
        d.month = 12 and d.year = 2008
     GROUP BY
        p.district_key) m12 ON c.district_key = m12.district_key
ORDER BY
    "Jan_2009" ASC
LIMIT 15;
```

```
 * postgresql://student@/FinalProj
15 rows affected.
```

Out[17]:

district	Jan_2009	Feb_2009	Mar_2009	April_2009	Sep_2008	Oct_2008	Nov_
CITY OF KINGSTON UPON HULL	84234.08	82144.08	87272.68	77555.97	95183.78	97815.69	1007
HYNDBURN	89998.81	88304.50	94238.89	98036.79	94598.01	103116.23	869
MERTHYR TYDFIL	94764.75	93825.00	105696.04	121737.04	104069.61	92028.84	1204
BLAENAU GWENT	96000.00	79972.04	76934.62	76956.72	85310.50	87671.60	917
BOLSOVER	96567.38	100960.24	119004.07	118910.22	131851.82	118508.57	1263
STOKE-ON-TRENT	96608.48	84849.92	88870.68	100842.14	106667.81	98053.52	981
OLDHAM	96860.89	130206.07	115134.05	122651.13	149665.46	134482.82	1227
CAERPHILLY	97944.80	116750.78	115897.58	113509.55	117431.34	119010.86	1201
HARTLEPOOL	98057.67	84626.14	106083.80	96647.46	114374.05	140821.86	1073
ASHFIELD	98423.92	114141.02	107598.93	110263.69	119748.61	117585.65	1130
BURNLEY	99626.08	75017.71	82333.04	89124.93	87793.13	95602.08	865
BLACKPOOL	99853.19	117266.45	107160.77	111863.65	126044.56	110138.81	1196
RHONDDA CYNON TAFF	101543.39	102419.64	104826.00	108063.44	101608.32	102812.65	936
BARROW-IN-FURNESS	102884.09	113354.15	97038.24	105512.88	116970.41	111825.93	1113
MANSFIELD	103048.97	111053.02	116692.94	112155.36	131332.41	128103.92	1110

What are the highest sales prices per month for the town city of LONDON for all years?

In [ ]:

In [192…

```
%%sql
SELECT
    d.year,
    d.month,
    ROUND(sum(p.price)) AS sum_prices,
    COUNT(p.*) AS number_of_sales,
    ROUND(SUM(p.price) / COUNT(p.*)) AS average_price_per_sale

FROM
    property_sales p
JOIN
    date_dimension d ON p.date_key = d.date_key
JOIN
    towncity_dimension t ON p.towncity_key = t.towncity_key
WHERE
    t.towncity = 'LONDON'
GROUP BY
    d.year,
    d.month
ORDER BY
    sum_prices DESC
    Limit 10;
```

```
 * postgresql://student@/FinalProj
10 rows affected.
```

Out[192]:

year	month	sum_prices	number_of_sales	average_price_per_sale
2007	7	4221486797	9648	437550
2007	8	4158778819	9877	421057
2007	6	3765645180	9201	409265
2006	8	3687538639	10192	361807
2006	7	3622205276	9837	368223
2006	9	3495646897	9468	369206
2007	9	3463064119	8219	421349
2006	6	3423223633	9495	360529
2007	3	3369982246	8751	385097
2006	12	3298585462	9021	365656

Now can check the lowest time, and we can see the real impact of 2008 is between the last quarter of 2008 and the beginning of the 2009 quarter

In [190…

```
%%sql
SELECT
    d.year,
    d.month,
    ROUND(sum(p.price)) AS sum_prices,
    COUNT(p.*) AS number_of_sales,
    ROUND(SUM(p.price) / COUNT(p.*)) AS average_price_per_sale

FROM
    property_sales p
JOIN
    date_dimension d ON p.date_key = d.date_key
JOIN
    towncity_dimension t ON p.towncity_key = t.towncity_key
WHERE
    t.towncity = 'LONDON'
GROUP BY
    d.year,
    d.month
ORDER BY
    sum_prices
    Limit 10;
```

```
 * postgresql://student@/FinalProj
10 rows affected.
```

Out[190]:

year	month	sum_prices	number_of_sales	average_price_per_sale
2009	1	774629298	1867	414906
2009	2	784755248	1930	406609
2008	11	964004376	2364	407785
2009	3	972433628	2420	401832
2008	12	1007005905	2607	386270
2009	4	1038433666	2554	406591
2009	5	1228950165	2957	415607
2008	10	1271232246	3092	411136
2008	9	1376241809	3052	450931
2008	8	1496188528	3569	419218

In [ ]:

In [ ]: