Energy efficiency: BigQuery SQL queries

In order to query a large dataset, it has to be first uploaded to Google Cloud Storage, and then imported into BigQuery as a table. This document outlines the queries used for the story *Six out of 10 renters live in energy-inefficient homes*.

Queries list

Data check: counting the rows

The first attempt to import the data to BigQuery resulted in some rows being missed: this SQL query on the table (**certsjustrating**):

```
SELECT COUNT(*) FROM `datacamp-287416.tweetsatmps_month1.certsjustrating`
```

Returned a count of 20,125,217 rows - but checking the CSV file on the command line (`wc cersjustrating.csv`) returned 23,597,326 rows.

This problem showed itself when trying to count by year. This query doesn't return any results for 2021 or 22:

```
#Date functions docs at
#https://cloud.google.com/bigquery/docs/reference/standard-sql/date_functions
SELECT EXTRACT(YEAR FROM INSPECTION_DATE) as insp_yr, count(*) as certs_count
FROM `datacamp-287416.tweetsatmps_month1.certsjustrating`
GROUP BY insp_yr
ORDER BY insp_yr DESC
```

I imported the CSV again, but this time instead of using auto-detect for the schema, manually entered the fields — specifying the INSPECTION_DATE field as a STRING rather than date.

Now this SQL query on the new table, named **EPC_justratings**:

```
SELECT COUNT(*) FROM `datacamp-287416.tweetsatmps_month1.EPC_justratings`
```

Returns 23597325 - one fewer row because the header row is also counted by the wc command.

Count certificates by year

Because we've stored the inspection dates as strings, we can't use EXTRACT to grab the year - instead we use LEFT:

```
SELECT LEFT (INSPECTION_DATE, 4) AS insp_yr, count(*) AS certcount FROM
`datacamp-287416.tweetsatmps_month1.EPC_justratings`
GROUP BY insp_yr
ORDER BY insp_yr DESC
```

Count certificates >2018 by year, LA, rating and tenure

We want to export a dataset that can be used to look at change in the last 5 years by LA and tenure.

```
SELECT LEFT(INSPECTION_DATE, 4) AS insp_yr, CURRENT_ENERGY_RATING,
LOCAL_AUTHORITY_LABEL, TENURE, count(*) as certs_count FROM
`datacamp-287416.tweetsatmps_month1.EPC_justratings`
WHERE LEFT(INSPECTION_DATE, 4) = "2018"
 OR LEFT(INSPECTION_DATE, 4) = "2019"
 OR LEFT(INSPECTION_DATE, 4) = "2020"
 OR LEFT(INSPECTION_DATE, 4) = "2021"
 OR LEFT(INSPECTION_DATE, 4) = "2022"
GROUP BY insp_yr, LOCAL_AUTHORITY_LABEL, TENURE, CURRENT_ENERGY_RATING
ORDER BY insp_yr DESC, LOCAL_AUTHORITY_LABEL ASC, CURRENT_ENERGY_RATING ASC
```

Add year field to ‘justratings’ table — filter out NA

We want to expand our analysis to do rolling 5 year figures, so need year in a separate column as a number. We need to filter out NA or we get an error from the CAST not being able to convert that to an integer.

```
SELECT *, CAST(LEFT (INSPECTION_DATE, 4) AS INT) AS insp_yr FROM
`datacamp-287416.tweetsatmps_month1.EPC_justratings`
WHERE INSPECTION_DATE != "NA"
```

Count certs >2011 by year, LA, tenure

To do a 5 year rolling analysis we want 10 years of data. This generates too many records to export as a CSV, so [we filter out tenures we aren’t looking at - we also clean up that tenure field](#) along the way.

```
#This cleans up the TENURE field for Owner/owner and Rented/rental
SELECT insp_yr, CURRENT_ENERGY_RATING, LOCAL_AUTHORITY_LABEL,
REPLACE(LOWER(TENURE), "rented", "rental") AS tenureclean, count(*) as
certs_count
#We query the version where we added a year as a number
FROM `datacamp-287416.tweetsatmps_month1.EPC_justratings_plusYR`
#That allows this filter
WHERE insp_yr >2011
```

```
#We can also filter the parts of 'Owner/owner' and 'Rented/rental' to get it
below 90k results
AND (TENURE LIKE "%wner%" OR TENURE LIKE "%ent%")
GROUP BY insp_yr, LOCAL_AUTHORITY_LABEL, TENURE, CURRENT_ENERGY_RATING
ORDER BY insp_yr DESC, LOCAL_AUTHORITY_LABEL ASC, CURRENT_ENERGY_RATING ASC
```

ALL certificates: counting the rows

To analyse other columns in the original data, we have combined all 342 CSV files, and all 92 columns, into one large CSV which has then been uploaded to Google Cloud and imported to BigQuery. Before this resulted in some rows being missed so we check the total rows again using this query:

```
SELECT COUNT(*) FROM `datacamp-287416.tweetsatmps_month1.certificatescombined`
```

This returns a count of 23,993,844 rows. The original CSV file on the command line (`wc cersjustrating.csv`) returned 24,015,958 rows so we are only missing 0.1% of the full data (most likely due to date formats not being recognised in the import)

Comparing the import with the previous import (data up to quarter 3 2022) where dates were imported as strings, it looks like that 0.1% is pretty evenly distributed, apart from 2020, where 0.5% of rows were lost (we don't know how many are lost from 2022 as only overall totals were compared, and the Dec 22 dataset has 31% more rows from 2022).

```
#This cleans up the TENURE field for Owner/owner and Rented/rental
SELECT EXTRACT(YEAR FROM INSPECTION_DATE) AS insp_yr, CURRENT_ENERGY_RATING,
POTENTIAL_ENERGY_RATING, LOCAL_AUTHORITY_LABEL, REPLACE(LOWER(TENURE),
"rented", "rental") AS tenureclean, count(*) as certs_count
#We query the full dataset up to Q4 2022
FROM `datacamp-287416.tweetsatmps_month1.certificatescombined`
#INSPECTION_DATE was imported as datetime in that table so we need to use
EXTRACT
WHERE EXTRACT(YEAR FROM INSPECTION_DATE) >2011
#We can also filter the parts of 'Owner/owner' and 'Rented/rental'
AND TENURE LIKE "%wner%" OR TENURE LIKE "%ent%"
GROUP BY INSPECTION_DATE, LOCAL_AUTHORITY_LABEL, TENURE, CURRENT_ENERGY_RATING,
POTENTIAL_ENERGY_RATING
```

```
ORDER BY EXTRACT(YEAR FROM INSPECTION_DATE) DESC, LOCAL_AUTHORITY_LABEL ASC,
CURRENT_ENERGY_RATING ASC
```

Current vs potential energy rating

The results of that query (9m rows) are saved as a new BigQuery table called `EPCcurrentVpotential`.

We query that to find out how many properties cannot meet the C rating target.

```
#select the year plus current and potential energy rating, count totals
SELECT insp_yr, CURRENT_ENERGY_RATING, POTENTIAL_ENERGY_RATING, COUNT(*)
FROM `datacamp-287416.tweetsatmps_month1.EPCcurrentVpotential`
#filter to the last decade or so
WHERE insp_yr > 2011
GROUP BY insp_yr, CURRENT_ENERGY_RATING, POTENTIAL_ENERGY_RATING
ORDER BY insp_yr ASC
```

Current vs potential energy rating by tenure and LA

We can expand the query to include data on local authority and tenure.

```
#select the year, tenure, LA plus current and potential energy rating, count
totals
SELECT insp_yr, CURRENT_ENERGY_RATING, POTENTIAL_ENERGY_RATING,
LOCAL_AUTHORITY_LABEL, tenureclean, COUNT(*)
FROM `datacamp-287416.tweetsatmps_month1.EPCcurrentVpotential`
#filter to the last decade or so
WHERE insp_yr > 2011
GROUP BY insp_yr, LOCAL_AUTHORITY_LABEL, tenureclean, CURRENT_ENERGY_RATING,
POTENTIAL_ENERGY_RATING
ORDER BY insp_yr ASC
```

However, the resulting table is 218,000 rows - too large to export as a CSV.

We refine the query to categorise the individual ratings within A-C and D-G true/false buckets, then:

```
#we create new columns which say whether the current/potential ratings are A-C
or not
SELECT insp_yr, (CURRENT_ENERGY_RATING LIKE "C" OR CURRENT_ENERGY_RATING LIKE
"B" OR CURRENT_ENERGY_RATING LIKE "A") AS currentAtoC, (POTENTIAL_ENERGY_RATING
LIKE "C" OR POTENTIAL_ENERGY_RATING LIKE "B" OR POTENTIAL_ENERGY_RATING LIKE
"A") AS potAtoC, LOCAL_AUTHORITY_LABEL, tenureclean, COUNT(*)
FROM `datacamp-287416.tweetsatmps_month1.EPCcurrentVpotential`
#filter to the last decade or so
WHERE insp_yr > 2011
GROUP BY insp_yr, LOCAL_AUTHORITY_LABEL, tenureclean, currentAtoC, potAtoC
ORDER BY insp_yr ASC
```

This reduces the rows to 33,000, which we can export and analyse in a spreadsheet.

Current vs potential energy rating - private only

We can also generate tables to see how many properties can achieve each grade:

```
SELECT POTENTIAL_ENERGY_RATING, COUNT(*) AS inspections
FROM `datacamp-287416.tweetsatmps_month1.EPC_UPRN_2012PLUS`
#filter to private rental only and last 5 years
WHERE insp_yr > 2018 AND TENURE LIKE "%rivat%"
GROUP BY POTENTIAL_ENERGY_RATING
ORDER BY POTENTIAL_ENERGY_RATING ASC
```

Or do it like this:

```
SELECT POTENTIAL_ENERGY_RATING, SUM(certs_count) as certs_total, insp_yr,
LOCAL_AUTHORITY_LABEL, tenureclean FROM
`datacamp-287416.tweetsatmps_month1.EPCcurrentVpotential`
#filter to private rental only and 2012 on
WHERE insp_yr > 2011 AND tenureclean LIKE "%rivat%"
GROUP BY insp_yr, LOCAL_AUTHORITY_LABEL, POTENTIAL_ENERGY_RATING, tenureclean
ORDER BY LOCAL_AUTHORITY_LABEL DESC, insp_yr
```

Joining the recommendations and the certificates data tables

To look at the recommendations we need to join that data to the certificates, which contains data on tenure, year and local authority that isn't in the recommendations table. When we do that the first thing we do is count how many rows are in the resulting table:

```
SELECT COUNT(*)
#rename this table as a
FROM `datacamp-287416.tweetsatmps_month1.EPC_recscombined` a
#rename this as b
LEFT JOIN `datacamp-287416.tweetsatmps_month1.certificatescombined` b
#so it's easier to then name the fields we are joining on
ON (a.LMK_KEY = b.LMK_KEY)
```

We get 95,830,257.

The recommendations table has 95,830,257 rows - so that looks good.

Filtering the recommendations join

Next we want to filter it to 2012 onwards and certain columns:

```
SELECT a.LMK_KEY,
 a.INDICATIVE_COST,
 #Most have a £ but some don't so £* means 'none or more'
 #followed by none or more numbers: [0-9]*
 #followed by none or more commas: ,*
 #followed by one or more numbers: [0-9]+
 #remove £ and , and convert to a number
 CAST(REPLACE(REPLACE(REGEXP_EXTRACT(INDICATIVE_COST,
"£*[0-9]*,*[0-9]+"),"£",""),",","") AS INT) AS lowercost,
 #as above but with $ at the end to indicate 'at the end of the string'
 CAST(REPLACE(REPLACE(REGEXP_EXTRACT(INDICATIVE_COST,
"£*[0-9]*,*[0-9]+$"),"£",""),",","") AS INT) AS uppercost,
```

```
 a.IMPROVEMENT_DESCR_TEXT,
 a.IMPROVEMENT_ID,
 a.IMPROVEMENT_ID_TEXT,
 a.IMPROVEMENT_ITEM,
 a.IMPROVEMENT_SUMMARY_TEXT,
 EXTRACT(YEAR FROM b.INSPECTION_DATE) AS insp_yr,
 b.LOCAL_AUTHORITY_LABEL,
 b.TENURE,
 b.UPRN,
 b.POSTCODE,
 b.CURRENT_ENERGY_RATING,
 b.POTENTIAL_ENERGY_RATING,
 b.ENERGY_CONSUMPTION_CURRENT,
 b.ENERGY_CONSUMPTION_POTENTIAL,
 COUNT(*)
#rename this table as a
FROM `datacamp-287416.tweetsatmps_month1.EPC_recscombined` a
#rename this as b
LEFT JOIN `datacamp-287416.tweetsatmps_month1.certificatescombined` b
#so it's easier to then name the fields we are joining on
ON (a.LMK_KEY = b.LMK_KEY)
#filter years before 2012
WHERE EXTRACT(YEAR FROM b.INSPECTION_DATE) > 2011
GROUP BY a.LMK_KEY,
 a.INDICATIVE_COST,
 lowercost,
 uppercost,
 a.IMPROVEMENT_DESCR_TEXT,
 a.IMPROVEMENT_ID,
 a.IMPROVEMENT_ID_TEXT,
 a.IMPROVEMENT_ITEM,
 a.IMPROVEMENT_SUMMARY_TEXT,
 insp_yr,
 b.TENURE,
 b.UPRN,
 b.BUILDING_REFERENCE_NUMBER,
```

```
 b.POSTCODE,
 b.CURRENT_ENERGY_RATING,
 b.POTENTIAL_ENERGY_RATING,
 b.ENERGY_CONSUMPTION_CURRENT,
 b.ENERGY_CONSUMPTION_POTENTIAL,
 b.LOCAL_AUTHORITY_LABEL
ORDER BY insp_yr DESC,
 COUNT(*) ASC
```

We actually don't need the COUNT(*) parts because a subsequent ORDER BY that field shows that there are none higher than 0.

The resulting table is EPC_combinedMORE2012plus.

Revised: joining recs and inspections

Now our full SQL query looks like this:

```
#we use a. here and b. below to prefix the field because later on we rename
each table a and b
SELECT a.LMK_KEY,
 a.INDICATIVE_COST,
 #Most have a £ but some don't so £* means 'none or more'
 #followed by none or more numbers: [0-9]*
 #followed by none or more commas: ,*
 #followed by one or more numbers: [0-9]+
 #remove £ and , and convert to a number
 CAST(REPLACE(REPLACE(REGEXP_EXTRACT(INDICATIVE_COST,
"£*[0-9]*,*[0-9]+"),"£",""),",","") AS INT) AS lowercost,
 #as above but with $ at the end to indicate 'at the end of the string'
 CAST(CAST(REPLACE(REPLACE(REGEXP_EXTRACT(INDICATIVE_COST,
"£*[0-9]*,*[0-9]+\\.*[0-9]*$"),"£",""),",","") AS FLOAT64) AS INT) AS
uppercost,
 a.IMPROVEMENT_DESCR_TEXT,
 a.IMPROVEMENT_ID,
```

```
 a.IMPROVEMENT_ID_TEXT,
 a.IMPROVEMENT_ITEM,
 a.IMPROVEMENT_SUMMARY_TEXT,
 EXTRACT(YEAR FROM b.INSPECTION_DATE) AS insp_yr,
 #again, b. here is used to indicate the second table
 b.LOCAL_AUTHORITY_LABEL,
 b.TENURE,
 b.UPRN,
 b.POSTCODE,
 b.CURRENT_ENERGY_RATING,
 b.POTENTIAL_ENERGY_RATING,
 b.ENERGY_CONSUMPTION_CURRENT,
 b.ENERGY_CONSUMPTION_POTENTIAL,
 COUNT(*)
#this is where we rename this table as a
FROM `datacamp-287416.tweetsatmps_month1.EPC_recscombined` a
#rename this as b
LEFT JOIN `datacamp-287416.tweetsatmps_month1.certificatescombined` b
#so it's easier to then name the fields we are joining on
ON (a.LMK_KEY = b.LMK_KEY)
#filter years before 2012
WHERE EXTRACT(YEAR FROM b.INSPECTION_DATE) > 2011
#we have to name all the fields in the GROUP command
GROUP BY a.LMK_KEY,
 a.INDICATIVE_COST,
 lowercost,
 uppercost,
 a.IMPROVEMENT_DESCR_TEXT,
 a.IMPROVEMENT_ID,
 a.IMPROVEMENT_ID_TEXT,
 a.IMPROVEMENT_ITEM,
 a.IMPROVEMENT_SUMMARY_TEXT,
 insp_yr,
 b.TENURE,
 b.UPRN,
 b.BUILDING_REFERENCE_NUMBER,
```

```
 b.POSTCODE,
 b.CURRENT_ENERGY_RATING,
 b.POTENTIAL_ENERGY_RATING,
 b.ENERGY_CONSUMPTION_CURRENT,
 b.ENERGY_CONSUMPTION_POTENTIAL,
 b.LOCAL_AUTHORITY_LABEL
#put the most recent years top, and then the largest numbers of inspections
ORDER BY insp_yr DESC,
 COUNT(*) ASC
```

This is saved as a new table called ‘EPC_combinedwrecs2012plus’

Counting frequency of improvement recommendations by year

This gives us an idea of which improvements are most often recommended for which properties — but it generates half a million rows so cannot be downloaded as a CSV.

```
SELECT IMPROVEMENT_ID_TEXT, insp_yr, TENURE, LOCAL_AUTHORITY_LABEL, COUNT(*) AS
recstotal
FROM `datacamp-287416.tweetsatmps_month1.EPC_combinedwrecs2012plus`
GROUP BY IMPROVEMENT_ID_TEXT, LOCAL_AUTHORITY_LABEL, TENURE, insp_yr
ORDER BY insp_yr, recstotal DESC
```

How can we get it lower?

We have 60+ recommendations.

And we have 11 years, so that’s 660+ rows.

We have three main tenures, but also ‘no data’ and other categories - so what should be broadly 1800 rows comes out at over 1900. So we can try this which reduces it back to 1700 rows by cleaning different ways of referring to rental, and removing nulls.

```
SELECT IMPROVEMENT_ID_TEXT, insp_yr, REPLACE(LOWER(TENURE),"rented","rental")
AS tenureclean, COUNT(*) AS recstotal
FROM `datacamp-287416.tweetsatmps_month1.EPC_combinedwrecs2012plus`
WHERE TENURE IS NOT NULL
```

```
GROUP BY IMPROVEMENT_ID_TEXT, tenureclean, insp_yr
ORDER BY insp_yr, recstotal DESC
```

Counting frequency of improvement recommendations by tenure and LA

If we add a breakdown by LA too, it increases the rows to 420,000:

```
SELECT IMPROVEMENT_ID_TEXT, insp_yr, REPLACE(LOWER(TENURE),"rented","rental")
AS tenureclean, LOCAL_AUTHORITY_LABEL, COUNT(*) AS recstotal
FROM `datacamp-287416.tweetsatmps_month1.EPC_combinedwrecs2012plus`
WHERE TENURE IS NOT NULL
GROUP BY IMPROVEMENT_ID_TEXT, LOCAL_AUTHORITY_LABEL, tenureclean, insp_yr
ORDER BY insp_yr, recstotal DESC
```

We can reduce that to a manageable number by not breaking down by year, and instead generating a total for the last 5 years because we're only looking at scale.

```
SELECT IMPROVEMENT_ID_TEXT,
#because we're filtering to years after 2017 below we include that as a piece
of data
 "2018-22" AS insp_yr,
 #clean up the tenure col
 REPLACE(LOWER(TENURE),"rented","rental") AS tenureclean,
 LOCAL_AUTHORITY_LABEL, COUNT(*) AS recstotal
FROM `datacamp-287416.tweetsatmps_month1.EPC_combinedwrecs2012plus`
#filter out years before 2018, and null tenures
WHERE TENURE IS NOT NULL AND insp_yr >2017
GROUP BY IMPROVEMENT_ID_TEXT, LOCAL_AUTHORITY_LABEL, tenureclean, insp_yr
ORDER BY recstotal DESC
```

Portable heaters analysis

How often are they used?

```
SELECT SECONDHEAT_DESCRIPTION,
 #extract the year
```

```
 EXTRACT(YEAR FROM INSPECTION_DATE) as insp_yr,
 #clean the tenure
 REPLACE(LOWER(TENURE),"rented","rental") AS tenureclean,
 COUNT(*) as inspections
FROM `datacamp-287416.tweetsatmps_month1.certificatescombined`
#filter to 2012 on
WHERE EXTRACT(YEAR FROM INSPECTION_DATE) > 2011
GROUP BY SECONDHEAT_DESCRIPTION, tenureclean, insp_yr
ORDER BY inspections DESC
```

We can do a local breakdown, focusing on private property with this query:

```
SELECT SECONDHEAT_DESCRIPTION, LOCAL_AUTHORITY_LABEL,
#extract the year
EXTRACT(YEAR FROM INSPECTION_DATE) as insp_yr,
#clean the tenure
REPLACE(LOWER(TENURE),"rented","rental") AS tenureclean,
COUNT(*) as inspections
FROM `datacamp-287416.tweetsatmps_month1.certificatescombined`
#filter to 2012 on, and private only
WHERE EXTRACT(YEAR FROM INSPECTION_DATE) > 2011 AND
REPLACE(LOWER(TENURE),"rented","rental") LIKE "%rivat%"
GROUP BY SECONDHEAT_DESCRIPTION, LOCAL_AUTHORITY_LABEL, tenureclean, insp_yr
ORDER BY inspections DESC
```

Removing duplicate UPRNs where a property is inspected more than once

We are following the steps in this tutorial. First, we see how many duplicate entries we can identify:

```
SELECT UPRN, COUNT(*) as inspections
FROM `datacamp-287416.tweetsatmps_month1.certificatescombined`
GROUP BY UPRN
#filter to those with more than one entry
```

```
HAVING COUNT(*) >1
ORDER BY inspections DESC
```

This returns 4,969,274 UPRNs - including 'null' (no UPRN), which alone accounts for half a million (576,822) inspections.

If we export this as another BigQuery table, called '`EPC_UPRNcounts`', and query that:

```
SELECT SUM(inspections) FROM
`datacamp-287416.tweetsatmps_month1.EPC_UPRNcounts`
```

We get 11,694,453 inspections connected with properties that have had more than one inspection, or don't have a UPRN (null) - almost half the 23,993,844 inspections covered in the `certificatescombined` data.

A simpler approach is to group by UPRN:

```
SELECT UPRN, COUNT(*) as inspections
FROM `datacamp-287416.tweetsatmps_month1.certificatescombined`
GROUP BY UPRN
ORDER BY inspections DESC
```

This gives us 17,268,665 UPRNs.

Since 2012, 14,706,455:

```
SELECT UPRN, COUNT(*) as inspections
FROM `datacamp-287416.tweetsatmps_month1.certificatescombined`
#filter to those after 2011
WHERE EXTRACT(YEAR FROM INSPECTION_DATE) > 2011
GROUP BY UPRN
ORDER BY inspections DESC
```

Can we see the scale of multiple inspections by narrowing down to one year?

```
SELECT COUNT(*) as inspections
FROM `datacamp-287416.tweetsatmps_month1.certificatescombined`
#filter to those after 2011
WHERE EXTRACT(YEAR FROM INSPECTION_DATE) = 2022
AND UPRN IS NOT NULL
```

```
GROUP BY UPRN
ORDER BY inspections DESC
```

In 2022 it's 1,457,303 UPRNs. There were 1,701,753 inspections, but that also includes a lot of nulls. So taking those out we get:
1508641 inspections, of 1457302 UPRNs. (1508641-1457302)/1508641 = 0.03, or 3%.

What does that mean for a statement like "40% of properties were graded A-C"?

Well the most extreme case is that 3 of that 40% might be the double-inspected properties. That would make it 37 - but 37 out of 97, which works out at 38.2%.

Or 3 out of the 60% might be the double-inspected properties. That would make it 43 out of 97, or 44.3%.

We also need to check if the rate of duplication is higher in rental properties (which you'd expect to come on the market more often).

Scotland: cleaning and filtering the dataset

When the data is imported it doesn't fetch the column headings, so we need to rename those as we filter to the columns of interest. We also filter out rows where headers have been repeated during combination.

```
SELECT string_field_6 AS INSPECTION_DATE,
LEFT(string_field_6, 4) AS insp_yr,
string_field_14 AS CURRENT_ENERGY_RATING,
string_field_16 AS POTENTIAL_ENERGY_RATING,
string_field_22 AS IMPROVEMENTS,
string_field_83 AS LOCAL_AUTHORITY_LABEL,
string_field_41 AS SECONDHEAT_DESCRIPTION,
string_field_95 AS TENURE
FROM `datacamp-287416.tweetsatmps_month1.allepcSCOTLAND`
#filter out header row data
WHERE string_field_14 != "CURRENT_ENERGY_RATING"
AND string_field_16 != "Potential energy efficiency rating band"
ORDER BY LEFT(string_field_6, 4) ASC
```

Scotland: current energy rating by LA and year (private only)

Here's the query:

```
SELECT CURRENT_ENERGY_RATING, COUNT(*) as certs_total, insp_yr, LOCAL_AUTHORITY_LABEL,
TENURE
FROM `datacamp-287416.tweetsatmps_month1.EPC_scotland_cleaned`
WHERE CAST(insp_yr AS INT) > 2012 AND TENURE LIKE '%rivat%'
GROUP BY insp_yr, LOCAL_AUTHORITY_LABEL, CURRENT_ENERGY_RATING, TENURE
ORDER BY LOCAL_AUTHORITY_LABEL DESC, insp_yr
```

Scotland: potential energy rating by LA and year (private only)

Here's the query:

```
SELECT POTENTIAL_ENERGY_RATING, COUNT(*) as certs_total, insp_yr,
LOCAL_AUTHORITY_LABEL, TENURE
FROM `datacamp-287416.tweetsatmps_month1.EPC_scotland_cleaned`
WHERE CAST(insp_yr AS INT) > 2012 AND TENURE LIKE '%rivat%'
GROUP BY insp_yr, LOCAL_AUTHORITY_LABEL, POTENTIAL_ENERGY_RATING, TENURE
ORDER BY LOCAL_AUTHORITY_LABEL DESC, insp_yr
```

Scotland: portable heaters (private only)

Here's the query:

```
SELECT SECONDHEAT_DESCRIPTION, COUNT(*) as certs_total, insp_yr,
LOCAL_AUTHORITY_LABEL, TENURE
FROM `datacamp-287416.tweetsatmps_month1.EPC_scotland_cleaned`
#private rental only
WHERE CAST(insp_yr AS INT) > 2012 AND TENURE LIKE '%rivat%'
GROUP BY insp_yr, LOCAL_AUTHORITY_LABEL, SECONDHEAT_DESCRIPTION, TENURE
```

```
ORDER BY LOCAL_AUTHORITY_LABEL DESC, insp_yr
```

Scotland: SPLIT descriptions

The IMPROVEMENTS field in the dataset presents a problem:

- The field combines multiple improvements in the same cell. Each improvement is, however, separated by a pipe. Similarly, different elements of each improvement are prefixed, including:
    - Description:
    - Indicative Cost:
    - Typical Saving:
    - Energy Rating after improvement:
    - Environmental Rating after improvement:
    - Green Deal Eligible:
- The entries appear to be truncated after 300 characters.

Here's an example:

> *Description: Replace boiler with new condensing boiler; Indicative Cost: £2,200 - £3,000; Typical Saving: 54; Energy Rating after improvement: C 72; Environmental Rating after improvement: C 71; Green Deal Eligible: Y | Description: Floor insulation; Indicative Cost: £800 - £1,200; Typical Saving: 5…*

To address the second problem we would need to re-import the data, specifying a different data type in the schema. According to this StackOverflow thread, that would be TEXT or MEDIUMTEXT or LONGTEXT.

To address the first problem it looks like we need STRING_SPLIT as detailed here. However, BigQuery does not support that function, but it does have SPLIT, and a possible approach is detailed here. Documentation here, tutorial here.

This code works in splitting each improvement onto its own row, with the date only appearing once over empty cells.

```
SELECT INSPECTION_DATE,
#split on the pipe to create multiple rows
 SPLIT(IMPROVEMENTS, "|") as improvement
FROM `datacamp-287416.tweetsatmps_month1.EPC_scotland_improvements`
#add a limit while testing
```

```
LIMIT 10
```

It turns out that, having split in this way, the character limit is revealed to no longer apply. We have improvements beyond the 300 character limit.

Also, a REGEX function could work to extract from that to dedicated columns. We could do a query like this:

```
SELECT IMPROVEMENTS,
 #create new columns indicating if key words appear
 REGEXP_CONTAINS(IMPROVEMENTS, "oiler") AS boiler,
 REGEXP_CONTAINS(IMPROVEMENTS, "sulat") AS insulation
FROM `datacamp-287416.tweetsatmps_month1.EPC_scotland_improvements`
LIMIT 1
```

Scotland: descriptions private/2013+ only by LA

Let's expand that previous query:

```
SELECT INSPECTION_DATE, LOCAL_AUTHORITY_LABEL,
#split on the pipe to create multiple rows
 SPLIT(IMPROVEMENTS, "|") as improvement
FROM `datacamp-287416.tweetsatmps_month1.EPC_scotland_improvements`
#filter to 2018-22
WHERE CAST(insp_yr AS INT) > 2017
 #filter to private rental
 AND TENURE LIKE '%rivat%'
 #filter out null LA labels
 AND LOCAL_AUTHORITY_LABEL != ""
ORDER BY LOCAL_AUTHORITY_LABEL ASC
```

We store that as a new table. At this point we discover that the new column is an array column. To work with it we need to use OFFSET like this:

```
SELECT *, improvement[OFFSET(0)]
FROM `datacamp-287416.tweetsatmps_month1.EPC_scot_18-22improvements`
LIMIT 10
```

This selects the first improvement from each inspection. The guide to arrays on BigQuery provides some useful code we can adapt to 'flatten' the arrays https://cloud.google.com/bigquery/docs/reference/standard-sql/arrays

Here's the query

```
#select the field BEFORE we create it
SELECT LOCAL_AUTHORITY_LABEL, flattened_imp
FROM `datacamp-287416.tweetsatmps_month1.EPC_scot_18-22improvements`
#the improvement field is an array field, so we need to UNNEST it to split it
into different rows
CROSS JOIN UNNEST(improvement) AS flattened_imp
```

We can expand that using regex to create new columns

```
#select the field BEFORE we create it
SELECT LOCAL_AUTHORITY_LABEL, flattened_imp,
 #extract description, replacing marker text after
 REPLACE(REGEXP_EXTRACT(flattened_imp, "Description:.*; Ind"),"; Ind","") AS
description,
 #extract indicative cost, replacing marker text after
 REPLACE(REGEXP_EXTRACT(flattened_imp, "Indicative Cost:.*; Typ"),"; Typ","")
AS indicativecost,
 #extract green deal eligible, replacing marker text after
 REGEXP_EXTRACT(flattened_imp, "Green Deal Eligible:.*") AS greendealeligible,
FROM `datacamp-287416.tweetsatmps_month1.EPC_scot_18-22improvements`
#the improvement field is an array field, so we need to UNNEST it to split it
into different rows
CROSS JOIN UNNEST(improvement) AS flattened_imp
```

Now, saved as a new table, we can query that for aggregate data.