Workbook 2: What are the Distributions of Jobs by County and by Metropolitan/Micropolitan Area?

So far in Workbook 1, we've just looked at the data by census block, with each row representing one census block. We also have a geography crosswalk table, containing information about the census blocks, including information about county and zip codes. What if we wanted to look at the jobs dataset within the context of the county they were in? We'd need to somehow combine the information from these two datasets together. We do that using `JOIN` s.

Motivating Question

In the previous workbook, we explored a little bit of the Workplace Area Characteristic dataset as well as the Geography Crosswalk dataset. We were able to look at how jobs were in each census block, as well as how many census blocks there were in each county or metropolitan/micropolitan area. In this workbook, we're going to explore deeper into the data available to us to try to answer the question:

What are the characteristics of the distribution of jobs by county and by metropolitan/micropolitan area?

To answer this, we'll have to use data from multiple tables.

Joins

> Note: If a type of join is not specified then it is equivalent to an `inner_join` .

One of the nice things about relational databases is organization using multiple tables that are linked together in some way. For example, suppose we have one table with 6 rows called **Table A**:

blockid	C000
1	5
2	10
3	2
4	6
5	22
6	9

And another table with 5 rows called **Table B**:

blockid	CA01

blockid	CA01
2	2
5	4
6	1
7	2
8	0

Let's say we want to combine Table A and Table B so that we have one table that contains information about `blockid` , `C000` , and `CA01` . We want to do this by matching the two tables by what they have in common, `blockid` . That is, we want a table that looks like this (let's call this **Table C**):

blockid	C000	CA01
2	10	2
5	22	4
6	9	1

Table C has each `blockid` that was in both Table A and Table B. It also contains the appropriate values for `C000` and `CA01` corresponding to each `blockid` . This kind of matching can be quite tricky to figure out manually, since there are different numbers of rows in each table, not all of the `blockid` values match for the two tables, and there are some `blockid` values that aren't in both. Fortunately for us, SQL is well-equipped to handle this task using the `JOIN` statement.

SQL Code and how it works

Let's take a look at the SQL code that performs this join and break it down.

```
SELECT TOP 1000 * FROM ds_public_1.dbo.lodes_ca_wac_S000_JT00_2015
JOIN ds_public_1.dbo.lodes_ca_xwalk
ON lodes_ca_wac_S000_JT00_2015.w_geocode =
lodes_ca_xwalk.tabblk2010;
```

Let's look at the first two lines.

```
SELECT TOP 1000 * FROM ds_public_1.dbo.lodes_ca_wac_S000_JT00_2015
JOIN ds_public_1.dbo.lodes_ca_xwalk
```

Here, we want to `SELECT` each column from a data table that we get from joining the tables `lodes_ca_wac_S000_JT00_2015` and `lodes_ca_xwalk` . The second line takes the `lodes_ca_wac_S000_JT00_2015` table and joins the `lodes_ca_xwalk` table to it.

We can't just mash two tables together though -- we need some way of making sure that the appropriate rows match. We do this with the third line:

```
ON lodes_ca_wac_S000_JT00_2015.w_geocode =
lodes_ca_xwalk.tabblk2010
```

This part specifies what we're joining on. That is, what is the ID variable that is in both tables that we want to match. Notice that they don't need to be named the same in both tables, though you do need to specify what they are in each table, even if they are the same, as well as which table they are from.

If you run the full code below, you should see the first 1000 rows (because of the `TOP 1000`) of the joined table. You should be able to scroll through all of the variables and see that we've managed to merge the `lodes_ca_wac_S000_JT00_2015` and `lodes_ca_xwalk` tables together according to their census block IDs.

> **Side note:** We're only going to be displaying a few of the columns instead of using `SELECT *` like we showed above since we are only interested in using a few relevant columns to answer our question. Joining to get tables with many columns is perfectly fine, but we'll only look at a few at a time to also make it easier to follow in these exercises.

```
SELECT TOP 1000 wac.w_geocode, xwalk.tabblk2010, wac.c000,
wac.ca01, wac.ca02, wac.ca03, xwalk.ctyname, xwalk.cbsaname
FROM ds_public_1.dbo.lodes_ca_wac_S000_JT00_2015 wac
JOIN ds_public_1.dbo.lodes_ca_xwalk xwalk
ON wac.w_geocode = xwalk.tabblk2010;
```

Here, we've chosen to display the two census block ID variables we're joining on so that you can see the matching, as well as a few characteristics from each table. Notice that we've specified the table before each variable. That's generally only necessary when both tables have a column with the same name, but we've done it here for clarity. The following will do the exact same thing and run just fine since the two tables don't share any of the column names.

```
SELECT TOP 1000 w_geocode, tabblk2010, c000, ca01, ca02, ca03,
ctyname, cbsaname
FROM ds_public_1.dbo.lodes_ca_wac_S000_JT00_2015
JOIN ds_public_1.dbo.lodes_ca_xwalk
ON lodes_ca_wac_S000_JT00_2015.w_geocode =
lodes_ca_xwalk.tabblk2010;
```

Checkpoint: Get Census Data and Join with your Tables

Try joining the Residence Area Characteristics table with the Crosswalk table in a similar manner, as well as changing some of the columns to display. As you construct your query, make sure you answer the following question regarding the join:

- What are they being joined on? That is, what is the "ID" variable you're matching on in each table?
- What information does the resulting table give you? For example, after we join the WAC table to the geography crosswalk table, we are now able to determine counties or metropolitan/micropolitan areas for census block containing workplaces.

Different Types of Joins

We've so far done only one type of join, an inner join. This is the default join (which is why we didn't need to specify anything more in the code). However, there are different types of joins.

Left and Right Joins in SQL

Suppose we want to look at every single census block in one table, only filling in information from the second table if it exists. We'll illustrate this using Table A and Table B from before. Recall that our `JOIN` created Table C:

blockid	C000	CA01
2	10	2
5	22	4
6	9	1

Instead, we want to create the following table:

blockid	C000	CA01
1	5	*null*
2	10	2
3	2	*null*
4	6	*null*
5	22	4
6	9	1

Here, we've kept every single row in Table A, and simply filled in the information from Table B if it existed for that `blockid`. This is called a **LEFT JOIN**, since we're taking the table on the left (that is, Table A) and adding the information from Table B onto that. We could have also done a **RIGHT JOIN**, which does the same thing, except flipping the tables, giving us something that looks like:

blockid	C000	CA01
2	10	2
5	22	4
6	9	1

blockid	C000	CA01
7	*null*	2
8	*null*	0

Applying Left Joins

When might you use left or right joins? Suppose you want to know which census blocks don't have any jobs. Then, we'd want to make sure that we keep all of the census blocks in the geography crosswalk, even if they aren't present in the workplace area characteristics tables, and add in the workplace area characteristics. This would mean that any blocks with a `null` value in the `C000` column would be blocks without any jobs.

To do this `JOIN`, we can use the `LEFT JOIN` statement.

```
SELECT TOP 1000 w_geocode, tabblk2010, c000, ca01, ca02, ca03,
ctyname, cbsaname
FROM ds_public_1.dbo.lodes_ca_wac_S000_JT00_2015
LEFT JOIN ds_public_1.dbo.lodes_ca_xwalk
ON lodes_ca_wac_S000_JT00_2015.w_geocode =
lodes_ca_xwalk.tabblk2010;
```

This is very similar to what we've done already with `JOIN`, except we add the word `LEFT` to it. When doing `LEFT` and `RIGHT JOIN`s, make sure to keep track of which one is first. Here, `lodes.ca_wac` comes first, so that's the table on the "left" side.

Outer Join

An outer join keeps all unique ids, then puts `NULL` if it isn't part of that table. This is similar to a `LEFT` or `RIGHT JOIN`, except instead of only keeping all IDs from one table, it keeps them from both tables. Consider our example with Table A and Table B. We want to join them such that we get a table that looks like:

blockid	C000	CA01
1	5	*null*
2	10	2
3	2	*null*
4	6	*null*
5	22	4
6	9	1
7	*null*	2
8	*null*	0

In a way, it's like combining the `LEFT` and `RIGHT JOIN`s so that we have all information from both tables.

Applying Full Joins

Suppose we want to know which census blocks that contain either the residences of people with jobs in the state or the census blocks of the location of the workplace, but not both. We use `FULL JOIN` for that.

> Note: A `FULL JOIN` is the same thing as `outer_join` in R.

```
SELECT TOP 1000 *
FROM ds_public_1.dbo.lodes_ca_wac_S000_JT00_2015
FULL JOIN ds_public_1.dbo.lodes_ca_rac_S000_JT00_2015
ON lodes_ca_wac_S000_JT00_2015.w_geocode =
lodes_ca_rac_S000_JT00_2015.h_geocode;
```

This will let us see which census blocks contain values for both workplace characteristics and residence characteristics.

Checkpoint: Types of Joins

Consider the following situations. How would you answer the question posed? What type of join should you use for each one? Which tables do you need to join? Try doing the join.

- How many census blocks in the state contain a workplace and how many don't? Recall that the geography crosswalk table contains information about the census blocks in the state.
- Which county contains the most census blocks containing a workplace?
- Which metropolitan/micropolitan area had the most census blocks containing a residence of a worker in the state?

Using Joins With Aggregation Functions

Suppose we wanted to know the distribution of the total number of jobs in California by county. The employment information is in one table, and county information is in another table. We need to join them, then aggregate the jobs, by county. Even though this seems like a complicated multi-step process, we can actually do it all in one query. Let's break it down into two parts: the join and the aggregation.

The Join

We need to join the `lodes_ca_wac_S000_JT00_2015` table and the `lodes_ca_xwalk` table. Since we aren't worried about counties that have no jobs, we can do an inner join. Consider the following `JOIN` :

```
SELECT a.w_geocode, a.c000, b.ctyname
FROM ds_public_1.dbo.lodes_ca_wac_S000_JT00_2015 a
JOIN ds_public_1.dbo.lodes_ca_xwalk b
ON a.w_geocode = b.tabblk2010;
```

This should provide the `JOIN` that we want, as well as the relevant columns. We need to take the table we get from the join and apply the aggregation to it.

The Aggregation

From our joined table, we need `GROUP BY` county, then find the `SUM()` . For now, let's call our joined table " `joinedtable` " and write the query based on this table. Since we're also interested in what counties have the most jobs, we'll order by the sum in descending order.

```
SELECT TOP 1000 ctyname, SUM(cast(c000 as int))
FROM joinedtable
GROUP BY ctyname
ORDER BY SUM(cast(c000 as int)) DESC;
```

> Note: The variable `c000` is of type varchar (variable character) which means the `sum()` function cannot be performed on this type of field. In order to sum this column, we need to cast this variable as an `int` (integer) using the `cast()` function.

We've already figured out how to get table " `joinedtable` " using the `JOIN` s above. All we need to do is put the `JOIN` in.

```
SELECT TOP 1000 b.ctyname, SUM(cast(a.c000 as int))
FROM ds_public_1.dbo.lodes_ca_wac_S000_JT00_2015 a
JOIN ds_public_1.dbo.lodes_ca_xwalk b
ON a.w_geocode = b.tabblk2010
GROUP BY b.ctyname
ORDER BY SUM(cast(c000 as int)) DESC;
```

This gives us one long query that performs the `JOIN` , then aggregates in the way we want.

Checkpoint: Putting It All Together

Look back to the motivating question: **What are the characteristics of the distribution of jobs by county and by metropolitan/micropolitan area?**

Using what you know about joins and aggregation functions, try to answer the following questions:

- Which counties have the most jobs?
- Which counties do the workers live in the most?
- Which metropolitan/micropolitan areas have the most jobs?
- What are some summary statistics of jobs at the county or metropolitan/micropolitan level (e.g. average number of jobs per county)?