

---

# CSTDM09 - California Statewide Travel Demand Model

---

Model Development

Employment

Final System Documentation: Technical Note

---

ULTRANS  
Institute of Transportation  
Studies, UC Davis  
Davis, California

HBA Specto Incorporated  
Calgary, Alberta

May 2011

## Table of Contents

<b>1. Introduction .....</b>	<b>5</b>
<b>2. 2000 Employment.....</b>	<b>5</b>
2.1 Data Sources .....	5
2.2 Method of Synthesizing Employment.....	6
2.2.1 Overview of Process .....	6
2.2.2 Development of Zonal Targets .....	6
2.2.3 Development of Samples .....	7
2.2.4 Outputs.....	13
2.2.5 Synthetic Employment .....	14
<b>3. 2008 Employment.....</b>	<b>15</b>
3.1 Data Sources .....	15
3.1.1 Census Transportation Planning Package (CTPP) .....	15
3.1.2 OnTheMap (OTM).....	16
3.1.3 California Employment Development Department (EDD) .....	16
3.1.4 Industry by NAICS.....	17
3.2 Data Processing Method .....	19
3.2.1 Employment by Industry.....	19
3.2.2 Employment by Occupation .....	23
3.3. Results.....	24
<b>Appendix 1: SQL .....</b>	<b>25</b>
SQL1: Calculate Number of Employees by Industry in Each County. ....	25
SQL2: Calculate Number of Employees by Industry in Each County. ....	25
SQL3: Calculate the Percentage of Employees by Industry type for Each County in Each Occupation for 2000 .....	25
<b>Appendix 2: Python.....</b>	<b>26</b>

**Tables:**

**Table 1: Industry Categories for Employment Synthesis Samples ..... 9**

**Table 2: Occupation Categories for Employment Synthesis Samples ..... 10**

**Table 3: PECAS Activity Categories for Employment Synthesis Samples ..... 11**

**Table 4: Industry Crosswalk..... 18**

**Table 5: Example of OTM Adjustment: Nevada County ..... 21**

**Figures:**

**Figure 1: Snapshot of the Target Table for the Employment Synthesizer ..... 7**

**Figure 2: Snapshot of the Crosswalk Table for the Employment Synthesizer ..... 8**

**Figure 3: Raw Output from the Employment Synthesizer ..... 13**

**Figure 4: Final Output from the Employment Synthesizer Process ..... 14**

## **1. Introduction**

This document provides background information and a description of input employment for the California Statewide Travel Demand Model (CSTDM09) for the years 2000 and 2008. The data for the two years was developed separately because the full US Census data was only available in the year 2000 and thus other data had to be used for 2008.

In brief, the 2000 Census Transportation Planning Package (CTPP) provided suitable data to create targets for a synthetic dataset with all employees for 2000. The 2000 Public Use Microdata Sample (PUMS) from the census provided samples for the synthesis for each transportation analysis zone (TAZ). This synthetic dataset is a listing of real employees resampled into each TAZ to meet the targets for the synthesizer.

The 2008 data at times required the combination of multiple data sources. The product of the 2000 population synthesis uses the change in employment determined by the California Employment Development Department for each county and disaggregates this change to a TAZ level using the rates of change from the census Longitudinal and Household Dynamics (LEHD), OnTheMap 4 data products.

## **2. 2000 Employment**

### **2.1 Data Sources**

CTPP includes employment at the census block group level by industry and by occupation on separate tables, but not by industry and occupation combined. This leads to difficulty in allocating production and management workers. The definitions of industry are also coarse, with only the base 13 North American Industry Classification System (NAICS) categories provided, combining health and education, for instance.

InfoUSA employment data lists the number of employees of a company by geographic coordinates and a fine level NAICS code by which the employment totals by a finer industry breakdown can be derived for each TAZ.

## **2.2 Method of Synthesizing Employment**

The purpose of synthesizing employment in the year 2000 is to provide total employment by place of work by finer industry and occupation levels. The preparation of synthetic employment requires (1) samples, which are the job records to be used and (2) targets, which are the control totals for geographies in the model system (TAZs).

### **2.2.1 Overview of Process**

For the CSTDM09, employment is needed for workers by both industry and occupation. The industry categories describe the type of activity at a person's place of work, and the occupation categories describe the kind of work a person does to earn a living. For those industries with production and administrative workers, the two types of workers are separated by using information about occupation, which improves the model accuracy; for instance, a person going to see a movie is interested in going to the nearby cinema, not to Hollywood where the movie is actually produced. For information on industry, we used 13 NAICS categories. North American Industrial Classification System (NAICS) is the federal government's standard industry classification system that groups establishments into industries based on the activities in which they are primarily engaged. We used 24 SOC categories for information on occupation. The Standard Occupational Classification (SOC) system is the federal government's standard classification system for occupations. It groups occupations according to the nature of the work performed.

### **2.2.2 Development of Zonal Targets**

The first step in synthesizing employment is to create a target table which includes the employment totals by CTPP industry, by CTPP occupation, and by InfoUSA industry, reclassified to PECAS Activity at the TAZ level. These are done by aggregating CTPP data from the block groups to the tract level, and by assigning each company surveyed by InfoUSA a TAZ identification number (ID) in ArcGIS. In the target table, the weight of the employment totals from InfoUSA is 20% of that of CTPP, so they are used to inform, rather than control the census data. The target table is illustrated in Figure 1.

	A	B	C	D	E	F	W	X	Y	Z	AU	AV	AW	AX	
1	Industry						OCCUPATION				INFO USA				
2	TAZ	Agricultur	Mining	Utilities	Constructi	Manufact	managem	Farmers_f	buss_fin_	comput_n	InfoUSAT	az	AMUSEMEN	CATTLE R	CONSTRU
3	0	1	1	1	1	1	1	1	1	1	0	0.2	0.2	0.2	
4	100	20	0	16	80	45	235	0	75	0	100	51	0	6	
5	101	28	1	7	39	104	125	4	10	10	101	238	0	14	
6	102	49	1	1	30	35	65	4	25	0	102	3	45	6	
7	103	59	1	1	14	0	30	10	4	0	103	1	0	4	
8	104	4	0	0	14	0	35	0	0	0	104	51	0	6	
9	105	133	2	49	238	305	608	10	224	99	105	213	0	51	
10	106	30	0	18	129	54	169	0	61	0	106	46	0	15	
11	107	63	1	16	108	273	278	16	127	36	107	262	0	93	
12	108	390	0	31	169	477	404	25	132	74	108	166	0	100	
13	109	4	0	2	20	12	64	0	4	0	109	68	0	14	
14	110	4	0	0	54	8	85	0	12	0	110	204	0	26	
15	111	33	0	7	38	138	52	12	33	4	111	471	0	16	
16	112	24	0	16	71	35	83	0	49	35	112	3	0	7	
17	113	33	0	14	80	49	50	0	4	8	113	0	0	6	
18	114	30	0	7	25	105	70	20	15	25	114	3	0	0	
19	115	128	2	30	230	143	234	33	58	22	115	68	0	90	
20	116	55	0	16	8	889	70	15	8	4	116	0	0	10	
21	117	210	0	7	14	23	45	125	8	4	117	3	20	25	
22	118	10	0	8	80	34	75	10	10	0	118	83	2	15	
23	119	60	0	8	4	0	39	25	0	0	119	6	0	0	
24	120	8	0	9	88	18	198	0	149	28	120	31	0	13	
25	121	51	1	48	159	125	435	4	210	14	121	16	0	52	
26	122	35	4	7	0	0	8	28	0	0	122	0	0	0	
27	123	9	1	0	10	0	4	4	4	0	123	0	0	0	
28	124	62	7	3	18	84	100	14	19	8	124	86	0	0	

**Figure 1: Snapshot of the Target Table for the Employment Synthesizer**

### 2.2.3 Development of Samples

The second step is to build a crosswalk of CTPP industry, CTPP occupation, and PECAS Activity for each job. Jobs are taken from the number of employed persons in the U.S Census Public Use Microdata Sample (PUMS) dataset, which include detailed information about both the occupation and industry for 5% of the workers in California.

The employment synthesizer sample input is a look up table (also called a sample file) containing each employee, with a link to a specific *PumsSerialNumber*, *Pums-IndustryCode*, *PumsOccupationCode*, and *PECASActivityCode*. These codes provide lookup information to each of the corresponding tables to inform the synthesizer regarding which industry, occupation, and/or activity category corresponds to the job of interest. This is done in SQL Server. If the CSTDM is not being run in conjunction with CalSIM, then the *PECASActivityCode* is not used, and has no effect. This table along with the target table is used to run the employment synthesizer. The crosswalk table is illustrated in Figure 2.

	A	B	C	D	E	X	Y	Z	AA	AW	AX	AY	AZ	BA
1	personid	INDCEN	Agricultur	Mining	Utilities	OCCEN	managem	Farmers_f	buss_fin	PECASIND	AMUSEME	CATTLE_R	CONSTRU	CONSTRU
2	1428716	786	0	0	0	232	0	0	0	747	0	0	0	0
3	1428717	819	0	0	0	306	0	0	0	749	0	0	0	0
4	1428718	786	0	0	0	231	0	0	0	747	0	0	0	0
5	1428721	668	0	0	0	153	0	0	0	738	0	0	0	0
6	1428722	727	0	0	0	214	0	0	0	740	0	0	0	0
7	1428725	917	0	0	0	62	0	0	1	739	0	0	0	0
8	1428726	728	0	0	0	80	0	0	1	744	0	0	0	0
9	1428727	697	0	0	0	570	0	0	0	740	0	0	0	0
10	1428729	109	0	0	0	5	1	0	0	523	0	0	0	0
11	1428730	699	0	0	0	471	0	0	0	740	0	0	0	0
12	1428732	947	0	0	0	385	0	0	0	857	0	0	0	0
13	1428733	939	0	0	0	380	0	0	0	857	0	0	0	0
14	1428735	479	0	0	0	524	0	0	0	634	0	0	0	0
15	1428737	877	0	0	0	715	0	0	0	633	0	0	0	0
16	1428738	417	0	0	0	512	0	0	0	632	0	0	0	0
17	1428741	417	0	0	0	471	0	0	0	632	0	0	0	0
18	1428742	729	0	0	0	263	0	0	0	744	0	0	0	0
19	1428743	916	0	0	0	540	0	0	0	753	0	0	0	0
20	1428744	797	0	0	0	365	0	0	0	749	0	0	0	0
21	1428746	339	0	0	0	772	0	0	0	530	0	0	0	0
22	1428747	396	0	0	0	775	0	0	0	526	0	0	0	0
23	1428748	168	0	0	0	896	0	0	0	524	0	0	0	0

**Figure 2: Snapshot of the Crosswalk Table for the Employment Synthesizer**

The industry and occupation categories are listed in Tables 1 and 2 below.

**Table 1: Industry Categories for Employment Synthesis Samples**

<b>PUMS Industry Categories</b>
Agriculture
Mining
Utilities
Construction
Manufacturing
Wholesale
Retail
Transport
Information
Finance & Insurance
Real Estate
Professional, School and Technical Service
Management
Administration & Support
Education
Health Care
Arts, Entertainment & Recreation
Accommodation & Food Service
Other Service
Public Administration
Armed Forces

**Table 2: Occupation Categories for Employment Synthesis Samples**

<b>OCCEN ( <a href="#">Census Occupation Codes</a> )</b>
Management 1-16 & 22-43
Farm Managers 20-21
Business and Financial 50-95
Computer and Mathematical 100-124
Architecture and Engineering 130-156
Life, Physical and Social Science 160-196
Community and Social Service 200-206
Legal 210-215
Education, Training and Library 220 255
Arts, Design, Entertainment, Sports and Media 260-296
Healthcare Practitioners and Technical 300-354
Healthcare Support 360-365
Protective Service 370-395
Food Preparation and Serving 400-416
Building and Grounds Maintenance 420-425
Personal Care and Service 430-465
Sales and Related 470-496
Office and Administration 500-593
Farming, Fishing and Forestry 600-613
Construction and Extraction 620-694
Installation, Maintenance and Repair 700-762
Production 770-896
Transportation and Material Moving 900-975
Armed Forces 980-983

The PECAS Activity categories are shown in Table 3 below.

**Table 3: PECAS Activity Categories for Employment Synthesis Samples**

AMUSEMENT_SERVICES_AND_MEDIA_PRESENTATION_production
CATTLE_RANCHING_AND_FARMING_production
CONSTRUCTION_production
ELECTRIC_UTILITIES_production
FEDERAL_GOVERNMENT_Services_production
FIELD_AND_SEED_CROPS_FARMING_production
FINANCE_INSURANCE_LEGAL_production
FISHING_production
FORESTRY_AND_ALLIED_ACTIVITIES_production
FRUIT_AND_NUT_FARMING_production
GOVERNMENT_ENTERPRISES_production
GREENHOUSE_NURSERY_AND_FLORICULTURE_production
HEALTH_SERVICES_production
HOTELS_production
INDUSTRIAL_BASED_SERVICES_production
MANUFACTURING_Computers_Electronics_and_Electrical_production
MANUFACTURING_Food_production
MANUFACTURING_Machinery_and_Transportation_Equipment_production
MANUFACTURING_Metal_Steel_production
MANUFACTURING_Paper_Chemicals_Plastic_Rubber_Glass_Cement_production
MANUFACTURING_Petro-Chemicals_production
MANUFACTURING_Textiles_production
MEDIA_production
MILITARY_production
MINING_AND_EXTRACTION_production
NONCATTLE_ANIMAL_production
NON-ELECTRIC_UTILITIES_AND_COMMUNICATIONS_production
OFFICE_BASED_SERVICES_production
ONSITE_BUSINESS_SERVICES_production

ONSITE_PERSONAL_SERVICES_production
PRIMARY_K-12_EDUCATION_production
PROFESSIONAL_SERVICES_production
REAL_ESTATE_production
RELIGIOUS_SERVICES_production
RESTAURANTS_production
RETAIL_BASED_SERVICES_production
RETAIL_TRADE_Automotive_production
RETAIL_TRADE_General_Merchandise_production
RETAIL_TRADE_Unprepared_Food_production
SECONDARY_EDUCATION_production
STATE_AND_LOCAL_GOVT_Services_production
SUPPORT_ACTIVITIES_FOR_AGRICULTURE_AND_FORESTRY_production
TRANSPORTATION_SERVICES_production
VEGETABLE_AND_MELO_FARMING_production
WHOLESALE_TRADE_production

The categories used in the Tables 1 and 2 above are not exactly the categories that are given in the PUMS data; they are aggregated using ratios that are provided by the California Employment Development Department (EDD). The EDD provides [employment by industry data](#) (EID) and [archived agriculture employment data](#) (SIC). These datasets were used to appropriately separate the employment types provided by the pums records to match those of the target files. Once the categories for the sample files were determined, SQL code was used to lookup from each database to create the complete crosswalk table for all of the sectors.

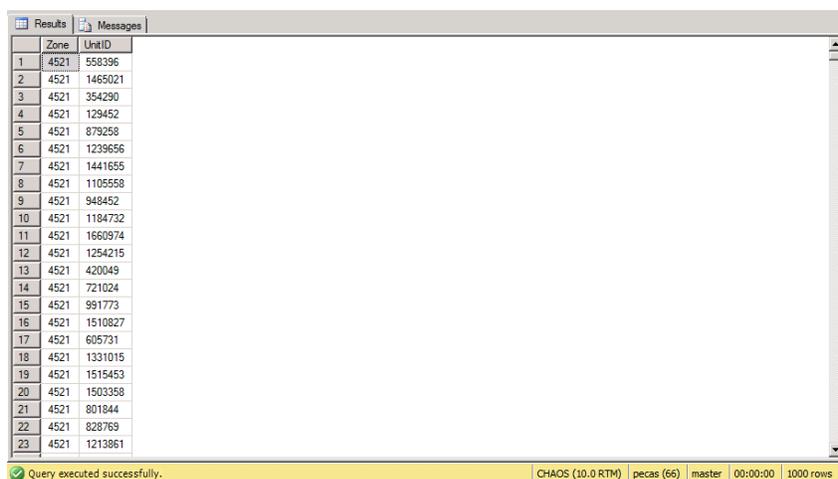
Finally, the employment synthesizer is run using the sample and target inputs. The software iteratively considers adding a unit from the sample to the list, subtracting a unit from the list, or 'swapping', in which a unit in the list is swapped out and a unit from the

sample is swapped in. The match of the list to the target values is scored using a goodness-of-fit function.

The list is divided into subgroups, which are commonly used to specify geographic areas known as “zones” which are to contain portions of a population. The process works through the list, subgroup by subgroup. For each subgroup, one of the three operations (add, subtract or swap) is selected with equal 1/3 probability. In the case of subtract or swap, a unit in the subgroup is randomly selected. In the case of add or swap, a unit in the sample is randomly selected. The operation is then performed, and the magnitude of the improvement in the goodness-of-fit score is calculated. If the goodness of fit improves the operation is kept. If the goodness of fit gets worse there is a less-than-1.0 probability that the operation will be kept, otherwise the operation will be undone.

### 2.2.4 Outputs

The output list of IDs is used to refer to the job information from the PUMS dataset. The synthesized database contains the number of employees at the place of work by detailed industry and occupation. Additionally, the outputs are processed using an SQL query to convert them into a more useful format to distinguish blue collar from white collar. The SQL reads the raw output file (see Figure 3) and places the numbered employees by industry and occupation into a table.



The screenshot shows a database query results window with a table containing 23 rows. The table has two columns: 'Zone' and 'UnitID'. The 'Zone' column contains the value '4521' for all rows. The 'UnitID' column contains a list of 23 unique integers ranging from 558396 to 1213861. The status bar at the bottom indicates 'Query executed successfully.' and '1000 rows'.

	Zone	UnitID
1	4521	558396
2	4521	1465021
3	4521	354290
4	4521	129452
5	4521	879258
6	4521	1239656
7	4521	1441655
8	4521	1105558
9	4521	948452
10	4521	1184732
11	4521	1660974
12	4521	1254215
13	4521	420049
14	4521	721024
15	4521	991773
16	4521	1510827
17	4521	605731
18	4521	1331015
19	4521	1515453
20	4521	1503358
21	4521	801844
22	4521	828769
23	4521	1213861

Figure 3: Raw Output from the Employment Synthesizer

### 2.2.5 Synthetic Employment

The output file includes the TAZ ID and the job ID which corresponds to the ID from the PUMS 5% dataset. These outputs contain detailed linked data of industry and occupation. The occupation is useful to separate the non-production workers from those production workers in some industries such as agriculture, construction, utilities, manufacturing, and retail; occupation is also an important component of the workplace location part of the Long Term Decision portion of the Short Distance Personal Travel Model. This detailed synthetic population of workers by industry and occupation, are aggregated to produce total counts of workers in each TAZ.

	Count_pid	PecasInd	PecasIndustryDescription	PecasOccCode	PecasOccName	Zone
1	1	0	NONE	101	Agriculture workers	3251
2	1	0	NONE	101	Agriculture workers	4282
3	1	0	NONE	101	Agriculture workers	5661
4	1	0	NONE	101	Agriculture workers	6027
5	2	0	NONE	102	Assembly and Fabrication workers	657
6	1	0	NONE	102	Assembly and Fabrication workers	895
7	1	0	NONE	102	Assembly and Fabrication workers	956
8	1	0	NONE	102	Assembly and Fabrication workers	1868
9	1	0	NONE	102	Assembly and Fabrication workers	2042
10	1	0	NONE	102	Assembly and Fabrication workers	2109
11	1	0	NONE	102	Assembly and Fabrication workers	3009
12	1	0	NONE	102	Assembly and Fabrication workers	3131
13	1	0	NONE	102	Assembly and Fabrication workers	3433
14	1	0	NONE	102	Assembly and Fabrication workers	4034
15	4	0	NONE	102	Assembly and Fabrication workers	4037
16	1	0	NONE	102	Assembly and Fabrication workers	4272
17	1	0	NONE	102	Assembly and Fabrication workers	4281
18	3	0	NONE	102	Assembly and Fabrication workers	4336
19	2	0	NONE	102	Assembly and Fabrication workers	4516
20	1	0	NONE	102	Assembly and Fabrication workers	4574
21	1	0	NONE	102	Assembly and Fabrication workers	5001
22	1	0	NONE	102	Assembly and Fabrication workers	5364
23	1	0	NONE	102	Assembly and Fabrication workers	5721
24	1	0	NONE	102	Assembly and Fabrication workers	5959
25	1	0	NONE	102	Assembly and Fabrication workers	6386
26	3	0	NONE	102	Assembly and Fabrication workers	6807
27	1	0	NONE	103	Business and financial operation...	455
28	1	0	NONE	103	Business and financial operation...	525
29	1	0	NONE	103	Business and financial operation...	882

Figure 4: Final Output from the Employment Synthesizer Process

### **3. 2008 Employment**

This document explains the process of developing a synthetic employment population for the year 2008 for validating the CSTDM09. To predict how individuals will travel within California, an accurate of the number of workers classified by industry and occupation in each TAZ is needed. Unfortunately, there was no single dataset available that met all the needs of the CSTDM09 for 2008 employment. A description of possible source datasets and the pros and cons of using them is detailed in the following section. However, a method for combining multiple datasets was developed to obtain 2008 employment target figures for each TAZ.

For the CSTDM09, employment is needed for workers by both industry and occupation. The industry categories describe the type of activity at a person's place of work, and the occupation categories describe the kind of work a person does to earn a living. For information on industry, we used 13 NAICS categories, which were aggregated to 9 industry categories used in the model. North American Industrial Classification System (NAICS) is the federal government's standard industry classification system that groups establishments into industries based on the activities in which they are primarily engaged. We used 24 SOC categories for information on occupation which were then aggregated to 9 occupation categories for the CSTDM. The Standard Occupational Classification (SOC) system is the federal government's standard classification system for occupations. It groups occupations according to the nature of the work performed.

#### **3.1 Data Sources**

##### ***3.1.1 Census Transportation Planning Package (CTPP)***

CTPP data come from the US census, and include the number of employees in each block group where they work by 14 NAICS categories. This dataset was used for the year 2000 employment synthesizer because it has a sufficient number of industry classes, has a small enough geography to aggregate to the model's TAZ system, and is considered a reliable and accurate count of employees where they work. Unfortunately,

this dataset is not available for the year 2008, but it is still used as our comparative base year dataset to calculate the amount of change from 2000 to 2008.

### **3.1.2 OnTheMap (OTM)**

OnTheMap data are a product of the Longitudinal Employment and Household Dynamics (LEHD) project of the UC Census Bureau. The LEHD combines federal and state administrative data on employers and employees with census data on where people live, to provide information on home-to-work flows. OnTheMap data are synthesized using Unemployment Insurance Wage Records reported by employers and maintained by each state for the purpose of administering its unemployment insurance system. Each state assigns employer locations, but actual business locations are not used in the dataset to retain the confidentiality of the workforce. Instead, the underlying data are modeled to produce a synthetic dataset which incorporates noise into the data to produce an accurate, but not exact, representation of employment.

The advantages of OnTheMap data are the geographic units it uses (i.e., census blocks), and the years of availability data (2002-2008). The disadvantages of using this dataset include the fact that it is a synthetic dataset and that several problems have been identified with the data, especially in the early years of publications (e.g., employees being linked to headquarters of companies rather than branch offices, which overestimates state workers in the state capital).

### **3.1.3 California Employment Development Department (EDD)**

The Division Labor Market Information (LMI) provides data to the public for the Employment Development Department on California labor markets. The QCEW, or Quarterly Census of Employment and Wages, release data by industry, including the number of employees in each industry for each county. The QCEW is a program involving the Bureau of Labor Statistics of the US Department of Labor and the State Employment Security Agencies (SESAs). Employment and wage information for workers is tabulated for all employees covered by state unemployment insurance (UI) laws and federal workers covered by the Unemployment Compensation for Federal

Employees (UCFE) program. At the State and area level, the QCEW program publishes employment and wage data down to the 6-digit NAICS industry level, if disclosure restrictions are met. In accordance with BLS policy, data provided to the bureau in confidence are not published and are used only for specified statistical purposes. BLS withholds publication of UI-covered employment and wage data for any industry level when necessary to protect the identity of cooperating employers.

The advantages of the EDD data are that they are based on real employment figures (not synthetically derived), they are published for the 13 industries required for our model, and they are available by month and/or quarter. One disadvantage of using EDD data is that they are available only to the disaggregate level of county, whereas we need data by TAZ. However, the data can be used as county-level control totals for each industry. Another disadvantage is that, in their commitment to protecting the confidentiality of the workforce, the EDD does not publish data for counties with low numbers of employees for certain industries. For example, Sierra County, 2008 population 3,343, has too few employees in six of the thirteen industry categories to meet the disclosure restrictions, and so those numbers are not reported. In some instances, other employment industries within the NAICS nesting structure were used when one of the thirteen industries had no information. For example, Sierra county did not disclose the number of employees working in NAICS industries 11 (Agriculture, Forestry, Fishing and Hunting) or 12 (Mining, Quarrying, and Oil and Gas Extraction), but they do disclose the number of employees for 11, 21 (Natural Resources and Mining). See Table 1 for a crosswalk table of NAICS codes and Industries.

### **3.1.4 Industry by NAICS**

We were able to evaluate each dataset using the most of the same industry categories, based off of NAICS. The single exception to this is for Military jobs. While CTPP 2000 provides total employment for the military, the other datasets (EDD and OTM) do not. For this reason, military employment in 2008 was assumed to be the same as 2000.

**Table 4: Industry Crosswalk**

NAICS	Description	CTPP Aggregation	EDD (most counties)	EDD (small counties)	OTM	CSTDM Categories
11	Agriculture, Forestry, Fishing and Hunting	Ag/Mining	Agriculture, Forestry, Fishing and Hunting	Natural Resources and Mining	Naics S01	Ag/Mining
21	Mining, Quarrying, and Oil and Gas Extraction		Mining	Natural Resources and Mining	Naics S02	
23	Construction	Construction	Construction	Construction	Naics S05	Construction
31-33	Manufacturing	Manufacturing	Manufacturing	Manufacturing	Naics S04	Manufacturing
42	Wholesale Trade	Wholesale	Wholesale Trade	*	Naics S06	Wholesale
22	Utilities	Trans / Util	Utilities	*	Naics S03	Trans / Util
48-49	Transportation and Warehousing		Transportation and Warehousing	*	Naics S08	
44-45	Retail Trade	Retail	Retail Trade	*	Naics S07	Retail
51	Information	Information	Information	Information	Naics S09	Information
52	Finance and Insurance	Finance/Ins/ RealEstate	Finance and Insurance	Financial Activities	Naics S10	Finance/Ins/ RealEstate
53	Real Estate and Rental and Leasing		Real Estate and Rental and Leasing		Naics S11	
54	Professional, Scientific, and Technical Services	Prof Sci/ Mgmt/Admin	Professional, Scientific, and Technical Services	Professional and Business Services	Naics S12	Prof Sci/ Mgmt/Admin
55	Management of Companies and Enterprises		Management of Companies and Enterprises		Naics S13	
56	Administrative and Support and Waste Management and Remediation Services	Prof Sci/ Mgmt/Admin	Administrative and Support and Waste Management and Remediation Services	Professional and Business Services	Naics S14	Prof Sci/ Mgmt/Admin
92	Public Administration	Government	Public Administration	Government	Naics S20	Government

61	Educational Services	Edu / Health	Education and Health Services	Education and Health Services	Naics S15	Edu / Health
62	Health Care and Social Assistance		Health Care and Social Assistance		Naics S16	
71	Arts, Entertainment, and Recreation	Arts/Rec/ Accom/Food	Arts, Entertainment, and Recreation	Leisure and Hospitality	Naics S17	Arts/Rec/ Accom/Food
72	Accommodation and Food Services			Leisure and Hospitality	Naics S18	
81	Other Services (except Public Administration)	Other Service	Other Services (except Public Administration)	Other Services	Naics S19	Other Service
xx	<i>Military employment, all industries</i>	<i>Armed Forces</i>	NA	NA	NA	NA

\* EDD provided data for Trade, Transportation and Utilities, but for the purposes of the model, trade could not be included with transportation and utilities

### 3.2 Data Processing Method

#### 3.2.1 Employment by Industry

To produce a reasonable estimation of 2008 employee counts by industry, we used the following method of combining the above data sources. Because the EDD data has employment counts by industry for each county for the years 2000 and 2008, we were able to obtain county-level targets of the growth or loss for each industry. We could then use the OTM data differences at the TAZ scale between 2002 and 2008 to allocate the county target levels to the TAZ level.

The EDD data for each industry was used as a countywide control total for the change in number of employees. This change was then disaggregated to TAZs using the share of the change for each TAZ in the county between the 2002 and 2008 OTM datasets as the disaggregation element. To accommodate both TAZs with positive and negative changes under OTM 2002 - 2008 in the same county, the TAZs with positive changes in

employment and negative changes in employment were addressed independently by scaling the net OTM change to match the EDD change.

**Table 5: Example of OTM Adjustment: Nevada County**

***Professional, Scientific and Technical Services/Management of Companies and Enterprises/Admin. and Support and Waste Mgmt and Remediation Services (Initial Condition)***

TAZ	CTPP		OTM		Adjust OTM		Final OTM		Change OTM	Calc Change	CTPP + Change
	2000	2002	2002	2008	2002	2008	2002	2008			
262	450	300	300	573			300	573	273	453	903
263	210	86	86	257			86	257	171	284	494
264	544	510	510	498			510	498	-12	-4	540
265	575	453	453	378			453	378	-75	-25	550
266	133	128	128	169			128	169	41	68	201
267	60	506	506	111			506	111	-395	-134	-74
268	350	101	101	102			101	102	1	2	352
269	530	222	222	405			222	405	183	304	834
270	295	105	105	197			105	197	92	153	448
271	89	50	50	66			50	66	16	27	116

**Professional, Scientific and Technical Services/Management of Companies and Enterprises/Admin. and Support and Waste Mgmt and Remediation Services (after adjustment)**

TAZ	CTPP			OTM		Adjust OTM		Final OTM		Change OTM	Calc Change	CTPP + Change
	2000	2002	2008	2002	2008	2002	2008	2002	2008			
262	450	300	573			300	573	273	425	875		
263	210	86	257			86	257	171	266	476		
264	544	510	498			510	498	-12	-5	539		
265	575	453	378			453	378	-75	-33	542		
266	133	128	169			128	169	41	64	197		
267	60	506	111	-200	100	306	211	-95	-42	18		
268	350	101	102			101	102	1	2	352		
269	530	222	405			222	405	183	285	815		
270	295	105	197			105	197	92	143	438		
271	89	50	66			50	66	16	25	114		

### **3.2.2 Employment by Occupation**

Because many industries have employees involved in many different job-related activities, it is important for the model to consider both industry and occupation. For example, Siemens Manufacturing Facility in Sacramento, CA makes light rail cars and equipment for rail services. Jobs for this company can include a Field Service Technician, who works on-site performing technical maintenance to rail cars and earns a yearly salary of \$40,000, as well as a Senior Civil Engineer, who likely works at the office headquarters, manages staff, requires a graduate degree and earns \$120,000 a year. Both employees work in the same industry, but have different daily tasks, different job locations, and different socioeconomic characteristics that affect the model.

Data sources for 2008 employment do not provide information on occupation. Instead, we used the 2000 Employment Synthesizer output to generate proportions of occupations for each industry for each TAZ. For example, the Retail Industry in TAZ 100 might have a distribution of:

- 80% Sales, Food and Entertainment
- 10% Managerial and Business
- 5% Professional and Technical
- 5% Clerical

TAZ 100 may also have an increase of 100 Retail Jobs between 2000 and 2008.

We would then speculate that TAZ 100 would have an increase for Retail of these occupations:

- +80 Sales, Food and Entertainment
- +10 Managerial and Business
- + 5 Professional and Technical
- + 5 Clerical

For each TAZ, the total change in the number of employees for each industry (2000-2008) was then distributed across 8 occupations: managerial and business; professional and technical; clerical; service (non-sales), health; education; sales, food and entertainment; blue collar; and military. This change in occupation was then added to the year 2000 occupation totals to create a 2008 occupation total dataset.

This process was automated through a set of queries executed in SQL Server using the 2000-2008 changes by industry for each TAZ, and the breakdown of occupation within each TAZ for each industry in 2000. This was then processed using python scripts to create an output table with the number of employees in each TAZ by both industry and occupation. See Appendices 1 and 2 for SQL and Python Code.

These outputs were then manually reviewed and cases with negative final numbers of employees by occupation were adjusted by reapportioning employees from high occupation totals to low. In the rare cases where occupation totals were less than zero, totals were adjusted so that the totals were greater than zero. In cases where, the 2000 occupation totals for a TAZ in Education, were large, totals for 2008 were checked to ensure that the representation of educational employment was maintained.

### **3.3. Results**

This process generated a set of employment numbers with both industry and occupation totals for each TAZ that were internally consistent.

## Appendix 1: SQL

### SQL1: Calculate Number of Employees by Industry in Each County.

```
SELECT TOP (100) PERCENT countyfp00, ctp_ind, SUM(tot) AS total
FROM (SELECT      dbo.CSTDM_EmpSum_CI.countyfp00, dbo.CSTDM_EmpSum_CI.ind,
      dbo.CSTDM_EmpSum_CI.tot, dbo.CSTDM_Ind_LUT.CTPP AS ctp_ind
FROM      dbo.CSTDM_EmpSum_CI LEFT OUTER JOIN
      dbo.CSTDM_Ind_LUT ON dbo.CSTDM_Ind_LUT.NAICS = dbo.CSTDM_EmpSum_CI.ind)
AS a
GROUP BY countyfp00, ctp_ind
ORDER BY countyfp00, ctp_ind
```

### SQL2: Calculate Number of Employees by Industry in Each County.

```
SELECT TOP (100) PERCENT countyfp00, ctp_ind, cstdm_occ, SUM(tot) AS total
FROM (SELECT a.countyfp00, a.ind, a.occ, a.tot, a.ctpp_ind, b.ShortName AS
      cstdm_occ
FROM (SELECT      dbo.CSTDM_EmpSum_CIO.countyfp00, dbo.CSTDM_EmpSum_CIO.ind,
      dbo.CSTDM_EmpSum_CIO.occ, dbo.CSTDM_EmpSum_CIO.tot,
      dbo.CSTDM_Ind_LUT.CTPP AS ctp_ind
FROM      dbo.CSTDM_EmpSum_CIO LEFT OUTER JOIN
      dbo.CSTDM_Ind_LUT ON dbo.CSTDM_Ind_LUT.NAICS = dbo.CSTDM_EmpSum_CIO.ind)
AS a INNER JOIN dbo.CSTDM_Occ_LUT AS b ON b.Code = a.occ) AS c
GROUP BY countyfp00, ctp_ind, cstdm_occ
ORDER BY countyfp00, ctp_ind, cstdm_occ
```

### SQL3: Calculate the Percentage of Employees by Industry type for Each County in Each Occupation for 2000

```
SELECT TOP (100) PERCENT a.countyfp00, a.ctpp_ind, a.cstdm_occ, a.total,
      b.total AS ind_tot, CAST(a.total AS float) / CAST(b.total AS float) AS perc
FROM      dbo.CSTDM_SumCIO AS a INNER JOIN dbo.CSTDM_SumCI AS b ON a.countyfp00 =
      b.countyfp00 AND a.ctpp_ind = b.ctpp_ind
ORDER BY a.countyfp00, a.ctpp_ind, a.cstdm_occ
```

## Appendix 2: Python

```
import pyodbc, csv, os, sys,datetime

print datetime.datetime.now(),"|Starting"
# Establish connection to SQL server
try:
    print datetime.datetime.now(),"|Opening connection"
    cnxn = pyodbc.connect('DRIVER={SQL
Server};SERVER=localhost;DATABASE=CAPUMS5;UID=pecas;PWD=icepecas')
    c1 = cnxn.cursor()
except Exception, e:
    print datetime.datetime.now(),"|Fatal Error"
    print e
    c1.close()
    cnxn.close()

# Create output table
try:
    # Check for existing table
    print datetime.datetime.now(),"|Checking for Table"
    if c1.tables(table='dbo.CSTDM_Emp08').fetchone():
        #Delete
        print datetime.datetime.now(),"|Deleting Existing Output Table"
        sqlstat = "DROP TABLE dbo.CSTDM_Emp08"

        #CreateTable
        print datetime.datetime.now(),"|Creating Output Table"
        crttable = "CREATE TABLE dbo.CSTDM_Emp08 (TAZ INTEGER, IN_EdHealth
integer, IN_LeisHosp integer, IN_Office integer, IN_Other integer, IN_PrimSec
integer, IN_Retail integer, IN_TransUtil integer, IN_Whole integer, OC_BluCol
integer, OC_Clerical integer, OC_Education integer, OC_Health integer,
OC_ManBus integer, OC_ProfTech integer, OC_SalesFE integer, OC_ServNS
integer)"
        c1.execute(crttable)
        cnxn.commit()
        print datetime.datetime.now(),"|Output Table Created"
        c1.close()
except Exception, e:
    print datetime.datetime.now(),"|Fatal Error"
    print e
    c1.close()
    cnxn.close()

# Main process
try:
    #set up main variables
    cTAZ = cnxn.cursor()

    #get TAZ loop
    cTAZ.execute("select taz, countyfp00 from dbo.CSTDM_CntyTAZ where taz <>0
order by taz")
    for rTAZ in cTAZ:
        TAZ = str(rTAZ[0])
        FIPS = str(rTAZ[1])
        print datetime.datetime.now(),"|Working on TAZ: " + str(TAZ)
```

```
# Reset Variables
dAgM = 0.0
dCon = 0.0
dEdH = 0.0
dFIRE = 0.0
dGovt = 0.0
dInf = 0.0
dLeis = 0.0
dMan = 0.0
dOther = 0.0
dPSA = 0.0
dRet = 0.0
dTU = 0.0
dWhole = 0.0
tAgM = 0.0
tCon = 0.0
tEdH = 0.0
tFIRE = 0.0
tGovt = 0.0
tInf = 0.0
tLeis = 0.0
tMan = 0.0
tOther = 0.0
tPSA = 0.0
tRet = 0.0
tTU = 0.0
tWhole = 0.0
iEdH = 0.0
iLH = 0.0
iOff = 0.0
iOther = 0.0
iPrimSec = 0.0
iRet = 0.0
iTU = 0.0
iWhole = 0.0
oBluCol = 0.0
oCler = 0.0
oEd = 0.0
oHealth = 0.0
oManBus = 0.0
oProfTech = 0.0
oSalesFE = 0.0
oServNS = 0.0
dBluCol = 0.0
dCler = 0.0
dEd = 0.0
dHealth = 0.0
dManBus = 0.0
dProfTech = 0.0
dSalesFE = 0.0
dServNS = 0.0
bBluCol = 0.0
bCler = 0.0
bEd = 0.0
bHealth = 0.0
bManBus = 0.0
```

```
bProfTech = 0.0
bSalesFE = 0.0
bServNS = 0.0

#get other rows
conAgM = pyodbc.connect('DRIVER={SQL
Server};SERVER=localhost;DATABASE=CAPUMS5;UID=pecas;PWD=icepecas')
cAgM = conAgM.cursor()
rAgM = cAgM.execute("select * from dbo.CSTDM_ind_AgMining where TAZ =
" + TAZ).fetchone()
cAgM.close()
conAgM.close()

conCon = pyodbc.connect('DRIVER={SQL
Server};SERVER=localhost;DATABASE=CAPUMS5;UID=pecas;PWD=icepecas')
cCon = conCon.cursor()
rCon = cCon.execute("select * from dbo.CSTDM_ind_Construction where
TAZ = " + TAZ).fetchone()
cCon.close()
conCon.close()

conEdH = pyodbc.connect('DRIVER={SQL
Server};SERVER=localhost;DATABASE=CAPUMS5;UID=pecas;PWD=icepecas')
cEdH = conEdH.cursor()
rEdH = cEdH.execute("select * from dbo.CSTDM_ind_EdHealth where TAZ =
" + TAZ).fetchone()
cEdH.close()
conEdH.close()

conFIRE = pyodbc.connect('DRIVER={SQL
Server};SERVER=localhost;DATABASE=CAPUMS5;UID=pecas;PWD=icepecas')
cFIRE = conFIRE.cursor()
rFIRE = cFIRE.execute("select * from dbo.CSTDM_ind_FIRE where TAZ = "
+ TAZ).fetchone()
cFIRE.close()
conFIRE.close()

conGovt = pyodbc.connect('DRIVER={SQL
Server};SERVER=localhost;DATABASE=CAPUMS5;UID=pecas;PWD=icepecas')
cGovt = conGovt.cursor()
rGovt = cGovt.execute("select * from dbo.CSTDM_ind_Govt where TAZ = "
+ TAZ).fetchone()
cGovt.close()
conGovt.close()

conInf = pyodbc.connect('DRIVER={SQL
Server};SERVER=localhost;DATABASE=CAPUMS5;UID=pecas;PWD=icepecas')
cInf = conInf.cursor()
rInf = cInf.execute("select * from dbo.CSTDM_ind_Information where
TAZ = " + TAZ).fetchone()
cInf.close()
conInf.close()

conLeis = pyodbc.connect('DRIVER={SQL
Server};SERVER=localhost;DATABASE=CAPUMS5;UID=pecas;PWD=icepecas')
```

```
        cLeis = conLeis.cursor()
        rLeis = cLeis.execute("select * from dbo.CSTDM_ind_Leisure where TAZ
= " + TAZ).fetchone()
        cLeis.close()
        conLeis.close()

        conMan = pyodbc.connect('DRIVER={SQL
Server};SERVER=localhost;DATABASE=CAPUMS5;UID=pecas;PWD=icepecas')
        cMan = conMan.cursor()
        rMan = cMan.execute("select * from dbo.CSTDM_ind_Manufacturing where
TAZ = " + TAZ).fetchone()
        cMan.close()
        conMan.close()

        conOther = pyodbc.connect('DRIVER={SQL
Server};SERVER=localhost;DATABASE=CAPUMS5;UID=pecas;PWD=icepecas')
        cOther = conOther.cursor()
        rOther = cOther.execute("select * from dbo.CSTDM_ind_Other where TAZ
= " + TAZ).fetchone()
        cOther.close()
        conOther.close()

        conPSA = pyodbc.connect('DRIVER={SQL
Server};SERVER=localhost;DATABASE=CAPUMS5;UID=pecas;PWD=icepecas')
        cPSA = conPSA.cursor()
        rPSA = cPSA.execute("select * from dbo.CSTDM_ind_ProfSciAdmin where
TAZ = " + TAZ).fetchone()
        cPSA.close()
        conPSA.close()

        conRet = pyodbc.connect('DRIVER={SQL
Server};SERVER=localhost;DATABASE=CAPUMS5;UID=pecas;PWD=icepecas')
        cRet = conRet.cursor()
        rRet = cRet.execute("select * from dbo.CSTDM_ind_Retail where TAZ = "
+ TAZ).fetchone()
        cRet.close()
        conRet.close()

        conTU = pyodbc.connect('DRIVER={SQL
Server};SERVER=localhost;DATABASE=CAPUMS5;UID=pecas;PWD=icepecas')
        cTU = conTU.cursor()
        rTU = cTU.execute("select * from dbo.CSTDM_ind_TransUtil where TAZ =
" + TAZ).fetchone()
        cTU.close()
        conTU.close()

        conWhole = pyodbc.connect('DRIVER={SQL
Server};SERVER=localhost;DATABASE=CAPUMS5;UID=pecas;PWD=icepecas')
        cWhole = conWhole.cursor()
        rWhole = cWhole.execute("select * from dbo.CSTDM_ind_Wholesale where
TAZ = " + TAZ).fetchone()
        cWhole.close()
        conWhole.close()
```

```
# Get industry changes
dAgM = float(rAgM[9])
dCon = float(rCon[9])
dEdH = float(rEdH[9])
dFIRE = float(rFIRE[9])
dGovt = float(rGovt[9])
dInf = float(rInf[9])
dLeis = float(rLeis[9])
dMan = float(rMan[9])
dOther = float(rOther[9])
dPSA = float(rPSA[9])
dRet = float(rRet[9])
dTU = float(rTU[9])
dWhole = float(rWhole[9])

# Get industry Totals
tAgM = int(rAgM[10])
tCon = int(rCon[10])
tEdH = int(rEdH[10])
tFIRE = int(rFIRE[10])
tGovt = int(rGovt[10])
tInf = int(rInf[10])
tLeis = int(rLeis[10])
tMan = int(rMan[10])
tOther = int(rOther[10])
tPSA = int(rPSA[10])
tRet = int(rRet[10])
tTU = int(rTU[10])
tWhole = int(rWhole[10])

# Group Industries
iEdH = tEdH
iLH = tLeis
iOff = tInf + tFIRE + tPSA + tGovt
iOther = tOther
iPrimSec = tAgM + tCon + tMan
iRet = tRet
iTU = tTU
iWhole = tWhole

# Get base Occupation Totals
conOcc = pyodbc.connect('DRIVER={SQL
Server};SERVER=localhost;DATABASE=CAPUMS5;UID=pecas;PWD=icepecas')
cOcc = conOcc.cursor()
cOcc.execute("select * from dbo.CSTDM_Occ00 where TAZ = " + TAZ)
for Occ in cOcc:
    bBluCol = Occ[1]
    bCler = Occ[2]
    bEd = Occ[3]
    bHealth = Occ[4]
    bManBus = Occ[5]
    bProfTech = Occ[7]
    bSalesFE = Occ[8]
    bServNS = Occ[6]
cOcc.close()
conOcc.close()
```

```
# Get perc table
conPerc = pyodbc.connect('DRIVER={SQL
Server};SERVER=localhost;DATABASE=CAPUMS5;UID=pecas;PWD=icepecas')
cPerc = conPerc.cursor()
cPerc.execute("SELECT * FROM dbo.CSTDM_Perc WHERE countyfp00 = '" +
FIPS + "'")
for perc in cPerc:
    base = 0.0
    if perc[1] == "Ag_Mining":
        #print "Processing: " + perc[1]
        base = float(dAgM)
        if perc[2] == "BluCol":
            #print perc[2] + ": " + str(dBluCol) + " + " + str(base)
+ " * " + str(perc[5])
            dBluCol = dBluCol + (base * float(perc[5]))
            #print "Equals " + str(dBluCol)
        elif perc[2] == "Clerical":
            #print perc[2] + ": " + str(dCler) + " + " + str(base) +
" * " + str(perc[5])
            dCler = dCler + (base * float(perc[5]))
            #print "Equals " + str(dCler)
        elif perc[2] == "Education":
            #print perc[2] + ": " + str(dEd) + " + " + str(base) + "
* " + str(perc[5])
            dEd = dEd + (base * float(perc[5]))
            #print "Equals " + str(dEd)
        elif perc[2] == "Health":
            #print perc[2] + ": " + str(dHealth) + " + " + str(base)
+ " * " + str(perc[5])
            dHealth = dHealth + (base * float(perc[5]))
            #print "Equals " + str(dHealth)
        elif perc[2] == "ManBus":
            #print perc[2] + ": " + str(dManBus) + " + " + str(base)
+ " * " + str(perc[5])
            dManBus = dManBus + (base * float(perc[5]))
            #print "Equals " + str(dManBus)
        elif perc[2] == "ProfTech":
            #print perc[2] + ": " + str(dProfTech) + " + " + str(base)
+ " * " + str(perc[5])
            dProfTech = dProfTech + (base * float(perc[5]))
            #print "Equals " + str(dProfTech)
        elif perc[2] == "SalesFE":
            #print perc[2] + ": " + str(dSalesFE) + " + " + str(base)
+ " * " + str(perc[5])
            dSalesFE = dSalesFE + (base * float(perc[5]))
            #print "Equals " + str(dSalesFE)
        elif perc[2] == "ServNS":
            #print perc[2] + ": " + str(dServNS) + " + " + str(base)
+ " * " + str(perc[5])
            dServNS = dServNS + (base * float(perc[5]))
            #print "Equals " + str(dServNS)
    elif perc[1] == "Arts Rec Accom Food":
        #print "Processing: " + perc[1]
        base = float(dLeis)
        if perc[2] == "BluCol":
```

```
        #print perc[2] + ": " + str(dBluCol) + " + " + str(base)
+ " * " + str(perc[5])
        dBluCol = dBluCol + (base * float(perc[5]))
        #print "Equals " + str(dBluCol)
        elif perc[2] == "Clerical":
        #print perc[2] + ": " + str(dCler) + " + " + str(base) +
" * " + str(perc[5])
        dCler = dCler + (base * float(perc[5]))
        #print "Equals " + str(dCler)
        elif perc[2] == "Education":
        #print perc[2] + ": " + str(dEd) + " + " + str(base) + "
* " + str(perc[5])
        dEd = dEd + (base * float(perc[5]))
        #print "Equals " + str(dEd)
        elif perc[2] == "Health":
+ " * " + str(perc[5])
        dHealth = dHealth + (base * float(perc[5]))
        #print "Equals " + str(dHealth)
        elif perc[2] == "ManBus":
+ " * " + str(perc[5])
        dManBus = dManBus + (base * float(perc[5]))
        #print "Equals " + str(dManBus)
        elif perc[2] == "ProfTech":
+ " * " + str(perc[5])
        dProfTech = dProfTech + (base * float(perc[5]))
        #print "Equals " + str(dProfTech)
        elif perc[2] == "SalesFE":
+ " * " + str(perc[5])
        dSalesFE = dSalesFE + (base * float(perc[5]))
        #print "Equals " + str(dSalesFE)
        elif perc[2] == "ServNS":
+ " * " + str(perc[5])
        dServNS = dServNS + (base * float(perc[5]))
        #print "Equals " + str(dServNS)
        elif perc[1] == "Construction":
        #print "Processing: " + perc[1]
        base = float(dCon)
        if perc[2] == "BluCol":
+ " * " + str(perc[5])
        dBluCol = dBluCol + (base * float(perc[5]))
        #print "Equals " + str(dBluCol)
        elif perc[2] == "Clerical":
        #print perc[2] + ": " + str(dCler) + " + " + str(base) +
" * " + str(perc[5])
        dCler = dCler + (base * float(perc[5]))
        #print "Equals " + str(dCler)
        elif perc[2] == "Education":
        #print perc[2] + ": " + str(dEd) + " + " + str(base) + "
* " + str(perc[5])
        dEd = dEd + (base * float(perc[5]))
        #print "Equals " + str(dEd)
```

```
        elif perc[2] == "Health":
            #print perc[2] + ": " + str(dHealth) + " + " + str(base)
+ " * " + str(perc[5])
            dHealth = dHealth + (base * float(perc[5]))
            #print "Equals " + str(dHealth)
        elif perc[2] == "ManBus":
            #print perc[2] + ": " + str(dManBus) + " + " + str(base)
+ " * " + str(perc[5])
            dManBus = dManBus + (base * float(perc[5]))
            #print "Equals " + str(dManBus)
        elif perc[2] == "ProfTech":
            #print perc[2] + ": " + str(dProfTech) + " + " + str(base)
+ " * " + str(perc[5])
            dProfTech = dProfTech + (base * float(perc[5]))
            #print "Equals " + str(dProfTech)
        elif perc[2] == "SalesFE":
            #print perc[2] + ": " + str(dSalesFE) + " + " + str(base)
+ " * " + str(perc[5])
            dSalesFE = dSalesFE + (base * float(perc[5]))
            #print "Equals " + str(dSalesFE)
        elif perc[2] == "ServNS":
            #print perc[2] + ": " + str(dServNS) + " + " + str(base)
+ " * " + str(perc[5])
            dServNS = dServNS + (base * float(perc[5]))
            #print "Equals " + str(dServNS)
        elif perc[1] == "Government":
            #print "Processing: " + perc[1]
            base = float(dGovt)
            if perc[2] == "BluCol":
                #print perc[2] + ": " + str(dBluCol) + " + " + str(base)
+ " * " + str(perc[5])
                dBluCol = dBluCol + (base * float(perc[5]))
                #print "Equals " + str(dBluCol)
            elif perc[2] == "Clerical":
                #print perc[2] + ": " + str(dCler) + " + " + str(base) +
" * " + str(perc[5])
                dCler = dCler + (base * float(perc[5]))
                #print "Equals " + str(dCler)
            elif perc[2] == "Education":
                #print perc[2] + ": " + str(dEd) + " + " + str(base) + "
* " + str(perc[5])
                dEd = dEd + (base * float(perc[5]))
                #print "Equals " + str(dEd)
            elif perc[2] == "Health":
                #print perc[2] + ": " + str(dHealth) + " + " + str(base)
+ " * " + str(perc[5])
                dHealth = dHealth + (base * float(perc[5]))
                #print "Equals " + str(dHealth)
            elif perc[2] == "ManBus":
                #print perc[2] + ": " + str(dManBus) + " + " + str(base)
+ " * " + str(perc[5])
                dManBus = dManBus + (base * float(perc[5]))
                #print "Equals " + str(dManBus)
            elif perc[2] == "ProfTech":
                #print perc[2] + ": " + str(dProfTech) + " + " + str(base)
+ " * " + str(perc[5])
                dProfTech = dProfTech + (base * float(perc[5]))
```

```
        #print "Equals " + str(dProfTech)
    elif perc[2] == "SalesFE":
        #print perc[2] + ": " + str(dSalesFE) + " + " + str(base)
+ " * " + str(perc[5])
        dSalesFE = dSalesFE + (base * float(perc[5]))
        #print "Equals " + str(dSalesFE)
    elif perc[2] == "ServNS":
        #print perc[2] + ": " + str(dServNS) + " + " + str(base)
+ " * " + str(perc[5])
        dServNS = dServNS + (base * float(perc[5]))
        #print "Equals " + str(dServNS)
    elif perc[1] == "Edu Health":
        #print "Processing: " + perc[1]
        base = float(dEdH)
        if perc[2] == "BluCol":
            #print perc[2] + ": " + str(dBluCol) + " + " + str(base)
+ " * " + str(perc[5])
            dBluCol = dBluCol + (base * float(perc[5]))
            #print "Equals " + str(dBluCol)
        elif perc[2] == "Clerical":
            #print perc[2] + ": " + str(dCler) + " + " + str(base) +
" * " + str(perc[5])
            dCler = dCler + (base * float(perc[5]))
            #print "Equals " + str(dCler)
        elif perc[2] == "Education":
            #print perc[2] + ": " + str(dEd) + " + " + str(base) + "
* " + str(perc[5])
            dEd = dEd + (base * float(perc[5]))
            #print "Equals " + str(dEd)
        elif perc[2] == "Health":
            #print perc[2] + ": " + str(dHealth) + " + " + str(base)
+ " * " + str(perc[5])
            dHealth = dHealth + (base * float(perc[5]))
            #print "Equals " + str(dHealth)
        elif perc[2] == "ManBus":
            #print perc[2] + ": " + str(dManBus) + " + " + str(base)
+ " * " + str(perc[5])
            dManBus = dManBus + (base * float(perc[5]))
            #print "Equals " + str(dManBus)
        elif perc[2] == "ProfTech":
            #print perc[2] + ": " + str(dProfTech) + " + " + str(base)
+ " * " + str(perc[5])
            dProfTech = dProfTech + (base * float(perc[5]))
            #print "Equals " + str(dProfTech)
        elif perc[2] == "SalesFE":
            #print perc[2] + ": " + str(dSalesFE) + " + " + str(base)
+ " * " + str(perc[5])
            dSalesFE = dSalesFE + (base * float(perc[5]))
            #print "Equals " + str(dSalesFE)
        elif perc[2] == "ServNS":
            #print perc[2] + ": " + str(dServNS) + " + " + str(base)
+ " * " + str(perc[5])
            dServNS = dServNS + (base * float(perc[5]))
            #print "Equals " + str(dServNS)
    elif perc[1] == "FIRE":
        #print "Processing: " + perc[1]
        base = float(dFIRE)
```

```
        if perc[2] == "BluCol":
            #print perc[2] + ": " + str(dBluCol) + " + " + str(base)
+ " * " + str(perc[5])
            dBluCol = dBluCol + (base * float(perc[5]))
            #print "Equals " + str(dBluCol)
        elif perc[2] == "Clerical":
            #print perc[2] + ": " + str(dCler) + " + " + str(base) +
" * " + str(perc[5])
            dCler = dCler + (base * float(perc[5]))
            #print "Equals " + str(dCler)
        elif perc[2] == "Education":
            #print perc[2] + ": " + str(dEd) + " + " + str(base) + "
* " + str(perc[5])
            dEd = dEd + (base * float(perc[5]))
            #print "Equals " + str(dEd)
        elif perc[2] == "Health":
+ " * " + str(perc[5])
            dHealth = dHealth + (base * float(perc[5]))
            #print "Equals " + str(dHealth)
        elif perc[2] == "ManBus":
+ " * " + str(perc[5])
            dManBus = dManBus + (base * float(perc[5]))
            #print "Equals " + str(dManBus)
        elif perc[2] == "ProfTech":
+ " * " + str(perc[5])
            dProfTech = dProfTech + (base * float(perc[5]))
            #print "Equals " + str(dProfTech)
        elif perc[2] == "SalesFE":
+ " * " + str(perc[5])
            dSalesFE = dSalesFE + (base * float(perc[5]))
            #print "Equals " + str(dSalesFE)
        elif perc[2] == "ServNS":
+ " * " + str(perc[5])
            dServNS = dServNS + (base * float(perc[5]))
            #print "Equals " + str(dServNS)
        elif perc[1] == "Information":
            #print "Processing: " + perc[1]
            base = float(dInf)
            if perc[2] == "BluCol":
+ " * " + str(perc[5])
                dBluCol = dBluCol + (base * float(perc[5]))
                #print "Equals " + str(dBluCol)
            elif perc[2] == "Clerical":
                #print perc[2] + ": " + str(dCler) + " + " + str(base) +
" * " + str(perc[5])
                dCler = dCler + (base * float(perc[5]))
                #print "Equals " + str(dCler)
            elif perc[2] == "Education":
                #print perc[2] + ": " + str(dEd) + " + " + str(base) + "
* " + str(perc[5])
                dEd = dEd + (base * float(perc[5]))
```

```
        #print "Equals " + str(dEd)
    elif perc[2] == "Health":
        #print perc[2] + ": " + str(dHealth) + " + " + str(base)
+ " * " + str(perc[5])
        dHealth = dHealth + (base * float(perc[5]))
        #print "Equals " + str(dHealth)
    elif perc[2] == "ManBus":
        #print perc[2] + ": " + str(dManBus) + " + " + str(base)
+ " * " + str(perc[5])
        dManBus = dManBus + (base * float(perc[5]))
        #print "Equals " + str(dManBus)
    elif perc[2] == "ProfTech":
        #print perc[2] + ": " + str(dProfTech) + " + " + str(base)
+ " * " + str(perc[5])
        dProfTech = dProfTech + (base * float(perc[5]))
        #print "Equals " + str(dProfTech)
    elif perc[2] == "SalesFE":
        #print perc[2] + ": " + str(dSalesFE) + " + " + str(base)
+ " * " + str(perc[5])
        dSalesFE = dSalesFE + (base * float(perc[5]))
        #print "Equals " + str(dSalesFE)
    elif perc[2] == "ServNS":
        #print perc[2] + ": " + str(dServNS) + " + " + str(base)
+ " * " + str(perc[5])
        dServNS = dServNS + (base * float(perc[5]))
        #print "Equals " + str(dServNS)
    elif perc[1] == "Manufacturing":
        #print "Processing: " + perc[1]
        base = float(dMan)
        if perc[2] == "BluCol":
            #print perc[2] + ": " + str(dBluCol) + " + " + str(base)
+ " * " + str(perc[5])
            dBluCol = dBluCol + (base * float(perc[5]))
            #print "Equals " + str(dBluCol)
        elif perc[2] == "Clerical":
            #print perc[2] + ": " + str(dCler) + " + " + str(base) +
" * " + str(perc[5])
            dCler = dCler + (base * float(perc[5]))
            #print "Equals " + str(dCler)
        elif perc[2] == "Education":
            #print perc[2] + ": " + str(dEd) + " + " + str(base) + "
* " + str(perc[5])
            dEd = dEd + (base * float(perc[5]))
            #print "Equals " + str(dEd)
        elif perc[2] == "Health":
            #print perc[2] + ": " + str(dHealth) + " + " + str(base)
+ " * " + str(perc[5])
            dHealth = dHealth + (base * float(perc[5]))
            #print "Equals " + str(dHealth)
        elif perc[2] == "ManBus":
            #print perc[2] + ": " + str(dManBus) + " + " + str(base)
+ " * " + str(perc[5])
            dManBus = dManBus + (base * float(perc[5]))
            #print "Equals " + str(dManBus)
        elif perc[2] == "ProfTech":
            #print perc[2] + ": " + str(dProfTech) + " + " + str(base)
+ " * " + str(perc[5])
```

```
        dProfTech = dProfTech + (base * float(perc[5]))
        #print "Equals " + str(dProfTech)
    elif perc[2] == "SalesFE":
        #print perc[2] + ": " + str(dSalesFE) + " + " + str(base)
+ " * " + str(perc[5])
        dSalesFE = dSalesFE + (base * float(perc[5]))
        #print "Equals " + str(dSalesFE)
    elif perc[2] == "ServNS":
        #print perc[2] + ": " + str(dServNS) + " + " + str(base)
+ " * " + str(perc[5])
        dServNS = dServNS + (base * float(perc[5]))
        #print "Equals " + str(dServNS)
    elif perc[1] == "Other Service":
        #print "Processing: " + perc[1]
        base = float(dOther)
        if perc[2] == "BluCol":
            #print perc[2] + ": " + str(dBluCol) + " + " + str(base)
+ " * " + str(perc[5])
            dBluCol = dBluCol + (base * float(perc[5]))
            #print "Equals " + str(dBluCol)
        elif perc[2] == "Clerical":
            #print perc[2] + ": " + str(dCler) + " + " + str(base) +
" * " + str(perc[5])
            dCler = dCler + (base * float(perc[5]))
            #print "Equals " + str(dCler)
        elif perc[2] == "Education":
            #print perc[2] + ": " + str(dEd) + " + " + str(base) + "
* " + str(perc[5])
            dEd = dEd + (base * float(perc[5]))
            #print "Equals " + str(dEd)
        elif perc[2] == "Health":
            #print perc[2] + ": " + str(dHealth) + " + " + str(base)
+ " * " + str(perc[5])
            dHealth = dHealth + (base * float(perc[5]))
            #print "Equals " + str(dHealth)
        elif perc[2] == "ManBus":
            #print perc[2] + ": " + str(dManBus) + " + " + str(base)
+ " * " + str(perc[5])
            dManBus = dManBus + (base * float(perc[5]))
            #print "Equals " + str(dManBus)
        elif perc[2] == "ProfTech":
            #print perc[2] + ": " + str(dProfTech) + " + " + str(base)
+ " * " + str(perc[5])
            dProfTech = dProfTech + (base * float(perc[5]))
            #print "Equals " + str(dProfTech)
        elif perc[2] == "SalesFE":
            #print perc[2] + ": " + str(dSalesFE) + " + " + str(base)
+ " * " + str(perc[5])
            dSalesFE = dSalesFE + (base * float(perc[5]))
            #print "Equals " + str(dSalesFE)
        elif perc[2] == "ServNS":
            #print perc[2] + ": " + str(dServNS) + " + " + str(base)
+ " * " + str(perc[5])
            dServNS = dServNS + (base * float(perc[5]))
            #print "Equals " + str(dServNS)
    elif perc[1] == "Prof Sci Admin":
        #print "Processing: " + perc[1]
```

```
        base = float(dPSA)
        if perc[2] == "BluCol":
            #print perc[2] + ": " + str(dBluCol) + " + " + str(base)
+ " * " + str(perc[5])
            dBluCol = dBluCol + (base * float(perc[5]))
            #print "Equals " + str(dBluCol)
        elif perc[2] == "Clerical":
            #print perc[2] + ": " + str(dCler) + " + " + str(base) +
" * " + str(perc[5])
            dCler = dCler + (base * float(perc[5]))
            #print "Equals " + str(dCler)
        elif perc[2] == "Education":
            #print perc[2] + ": " + str(dEd) + " + " + str(base) + "
* " + str(perc[5])
            dEd = dEd + (base * float(perc[5]))
            #print "Equals " + str(dEd)
        elif perc[2] == "Health":
            #print perc[2] + ": " + str(dHealth) + " + " + str(base)
+ " * " + str(perc[5])
            dHealth = dHealth + (base * float(perc[5]))
            #print "Equals " + str(dHealth)
        elif perc[2] == "ManBus":
            #print perc[2] + ": " + str(dManBus) + " + " + str(base)
+ " * " + str(perc[5])
            dManBus = dManBus + (base * float(perc[5]))
            #print "Equals " + str(dManBus)
        elif perc[2] == "ProfTech":
            #print perc[2] + ": " + str(dProfTech) + " + " + str(base)
+ " * " + str(perc[5])
            dProfTech = dProfTech + (base * float(perc[5]))
            #print "Equals " + str(dProfTech)
        elif perc[2] == "SalesFE":
            #print perc[2] + ": " + str(dSalesFE) + " + " + str(base)
+ " * " + str(perc[5])
            dSalesFE = dSalesFE + (base * float(perc[5]))
            #print "Equals " + str(dSalesFE)
        elif perc[2] == "ServNS":
            #print perc[2] + ": " + str(dServNS) + " + " + str(base)
+ " * " + str(perc[5])
            dServNS = dServNS + (base * float(perc[5]))
            #print "Equals " + str(dServNS)
        elif perc[1] == "Retail":
            #print "Processing: " + perc[1]
            base = float(dRet)
            if perc[2] == "BluCol":
                #print perc[2] + ": " + str(dBluCol) + " + " + str(base)
+ " * " + str(perc[5])
                dBluCol = dBluCol + (base * float(perc[5]))
                #print "Equals " + str(dBluCol)
            elif perc[2] == "Clerical":
                #print perc[2] + ": " + str(dCler) + " + " + str(base) +
" * " + str(perc[5])
                dCler = dCler + (base * float(perc[5]))
                #print "Equals " + str(dCler)
            elif perc[2] == "Education":
                #print perc[2] + ": " + str(dEd) + " + " + str(base) + "
* " + str(perc[5])
```

```
        dEd = dEd + (base * float(perc[5]))
        #print "Equals " + str(dEd)
    elif perc[2] == "Health":
        #print perc[2] + ": " + str(dHealth) + " + " + str(base)
+ " * " + str(perc[5])
        dHealth = dHealth + (base * float(perc[5]))
        #print "Equals " + str(dHealth)
    elif perc[2] == "ManBus":
        #print perc[2] + ": " + str(dManBus) + " + " + str(base)
+ " * " + str(perc[5])
        dManBus = dManBus + (base * float(perc[5]))
        #print "Equals " + str(dManBus)
    elif perc[2] == "ProfTech":
        #print perc[2] + ": " + str(dProfTech) + " + " + str(base)
+ " * " + str(perc[5])
        dProfTech = dProfTech + (base * float(perc[5]))
        #print "Equals " + str(dProfTech)
    elif perc[2] == "SalesFE":
        #print perc[2] + ": " + str(dSalesFE) + " + " + str(base)
+ " * " + str(perc[5])
        dSalesFE = dSalesFE + (base * float(perc[5]))
        #print "Equals " + str(dSalesFE)
    elif perc[2] == "ServNS":
        #print perc[2] + ": " + str(dServNS) + " + " + str(base)
+ " * " + str(perc[5])
        dServNS = dServNS + (base * float(perc[5]))
        #print "Equals " + str(dServNS)
    elif perc[1] == "Trans Util":
        #print "Processing: " + perc[1]
        base = float(dTU)
        if perc[2] == "BluCol":
            #print perc[2] + ": " + str(dBluCol) + " + " + str(base)
+ " * " + str(perc[5])
            dBluCol = dBluCol + (base * float(perc[5]))
            #print "Equals " + str(dBluCol)
        elif perc[2] == "Clerical":
            #print perc[2] + ": " + str(dCler) + " + " + str(base) +
" * " + str(perc[5])
            dCler = dCler + (base * float(perc[5]))
            #print "Equals " + str(dCler)
        elif perc[2] == "Education":
            #print perc[2] + ": " + str(dEd) + " + " + str(base) + "
* " + str(perc[5])
            dEd = dEd + (base * float(perc[5]))
            #print "Equals " + str(dEd)
    elif perc[2] == "Health":
        #print perc[2] + ": " + str(dHealth) + " + " + str(base)
+ " * " + str(perc[5])
        dHealth = dHealth + (base * float(perc[5]))
        #print "Equals " + str(dHealth)
    elif perc[2] == "ManBus":
        #print perc[2] + ": " + str(dManBus) + " + " + str(base)
+ " * " + str(perc[5])
        dManBus = dManBus + (base * float(perc[5]))
        #print "Equals " + str(dManBus)
    elif perc[2] == "ProfTech":
```

```
        #print perc[2] + ": " + str(dProfTech) + " + " + str(base)
+ " * " + str(perc[5])
        dProfTech = dProfTech + (base * float(perc[5]))
        #print "Equals " + str(dProfTech)
        elif perc[2] == "SalesFE":
        #print perc[2] + ": " + str(dSalesFE) + " + " + str(base)
+ " * " + str(perc[5])
        dSalesFE = dSalesFE + (base * float(perc[5]))
        #print "Equals " + str(dSalesFE)
        elif perc[2] == "ServNS":
        #print perc[2] + ": " + str(dServNS) + " + " + str(base)
+ " * " + str(perc[5])
        dServNS = dServNS + (base * float(perc[5]))
        #print "Equals " + str(dServNS)
        elif perc[1] == "Wholesale":
        #print "Processing: " + perc[1]
        base = float(dWhole)
        if perc[2] == "BluCol":
        #print perc[2] + ": " + str(dBluCol) + " + " + str(base)
+ " * " + str(perc[5])
        dBluCol = dBluCol + (base * float(perc[5]))
        #print "Equals " + str(dBluCol)
        elif perc[2] == "Clerical":
        #print perc[2] + ": " + str(dCler) + " + " + str(base) +
" * " + str(perc[5])
        dCler = dCler + (base * float(perc[5]))
        #print "Equals " + str(dCler)
        elif perc[2] == "Education":
        #print perc[2] + ": " + str(dEd) + " + " + str(base) + "
* " + str(perc[5])
        dEd = dEd + (base * float(perc[5]))
        #print "Equals " + str(dEd)
        elif perc[2] == "Health":
        #print perc[2] + ": " + str(dHealth) + " + " + str(base)
+ " * " + str(perc[5])
        dHealth = dHealth + (base * float(perc[5]))
        #print "Equals " + str(dHealth)
        elif perc[2] == "ManBus":
        #print perc[2] + ": " + str(dManBus) + " + " + str(base)
+ " * " + str(perc[5])
        dManBus = dManBus + (base * float(perc[5]))
        #print "Equals " + str(dManBus)
        elif perc[2] == "ProfTech":
        #print perc[2] + ": " + str(dProfTech) + " + " + str(base)
+ " * " + str(perc[5])
        dProfTech = dProfTech + (base * float(perc[5]))
        #print "Equals " + str(dProfTech)
        elif perc[2] == "SalesFE":
        #print perc[2] + ": " + str(dSalesFE) + " + " + str(base)
+ " * " + str(perc[5])
        dSalesFE = dSalesFE + (base * float(perc[5]))
        #print "Equals " + str(dSalesFE)
        elif perc[2] == "ServNS":
        #print perc[2] + ": " + str(dServNS) + " + " + str(base)
+ " * " + str(perc[5])
        dServNS = dServNS + (base * float(perc[5]))
        #print "Equals " + str(dServNS)
```

```
cPerc.close()
conPerc.close()

# assemble final values
oBluCol = int(round(bBluCol + dBluCol))
oCler = int(round(bCler + dCler))
oEd = int(round(bEd + dEd))
oHealth = int(round(bHealth + dHealth))
oManBus = int(round(bManBus + dManBus))
oProfTech = int(round(bProfTech + dProfTech))
oSalesFE = int(round(bSalesFE + dSalesFE))
oServNS = int(round(bServNS + dServNS))

insstr = "insert into dbo.CSTDM_Emp08(TAZ,IN_EdHealth,IN_LeisHosp,
IN_Office, IN_Other, IN_PrimSec, IN_Retail, IN_TransUtil, IN_Whole, OC_BluCol,
OC_Clerical, OC_Education, OC_Health, OC_ManBus,OC_ProfTech, OC_SalesFE,
OC_ServNS) values (" + str(TAZ) + "," + str(iEdH) + "," + str(iLH) + "," +
str(iOff) + "," + str(iOther) + "," + str(iPrimSec) + "," + str(iRet) + "," +
str(iTU) + "," + str(iWhole) + "," + str(oBluCol) + "," + str(oCler) + "," +
str(oEd) + "," + str(oHealth) + "," + str(oManBus) + "," + str(oProfTech) +
"," + str(oSalesFE) + "," + str(oServNS) + ")"
#print insstr
conIns = pyodbc.connect('DRIVER={SQL
Server};SERVER=localhost;DATABASE=CAPUMS5;UID=pecas;PWD=icepecas')
cIns = conIns.cursor()
cIns.execute(insstr)
conIns.commit()
cIns.close()
conIns.close()

except Exception, e:
    print datetime.datetime.now(),"|Fatal Error"
    print e
    cl.close()
    cnxn.close()

print datetime.datetime.now(),"|Process Complete"
```