FINAL REPORT

FOR H2B VISA APPLICATIONS

CASE SCENARIO

Our client is US labor department. The United States Department of Labor (DOL) is a cabinet-level department of the U.S. federal government responsible for occupational safety, wage and hour standards, unemployment insurance benefits, reemployment services, and some economic statistics, which has massive data. Currently, the DOL would like to investigate H2B applications to get insight of the demands of temporary non-agricultural jobs in US. Unlike H1B visa, H2B visa only permits foreign workers to come temporarily to the United States and perform temporary non-agricultural services or labor on a one-time, seasonal, peak-load or intermittent basis, which is more complex and difficult. All data directly from government official website can help the department to better understand the application market and applicants. We are built by the DOL from different department as a Data Analytics Task Team to design the database and manage data so that the most value can be made out of the H2B visa dataset.

Business leaders and members of both parties have long sought to expand the H-2B program because it avoids the need for these employers to compete for available U.S. workers. However, reports have shown that H2B jobs distort the labor market, lower wages and prevent those neediest Americans to find jobs. As a result, analyzing the H2B applications might be necessary for DOL to investigate the H2B status and compare it with unemployment status in US.

Also, this dataset has enough attributes for us to restructure the relational database and meet the requirement of 3NF tables during the normalization process. We would like to put ourselves into government's perspectives to see what is the application market like and what are common features of those applicants who got H2B visa.

To solve the issue, we have done investigations on work visa application process to have a basic understanding on H2B visa. We also examined the H2B visa dataset to ensure we have enough data and variable for the analysis. We plan to first exploratory data analysis, then do normalization. And the next step is to extract, transform and load data from one source into another database. Through MetaData, develop an interactive analytical dashboard to present all the results from both data and reality perspectives.

Since we are hired by the US government to analyze the H2B data and provide insightful conclusions for them to understand the application information and make corresponding decisions in the future, the project results are valuable to DOL, but also anyone who is interested in applying for H2B visa.

PROPOSAL

A sample of the data is shown as below (figure 1). The first eight columns of the dataset present the different features of H2B visa application, such as case number, decision data, case status and etc.

	CASE_NUMBER	DECISION_DATE	VISA_TYPE	SUBMITTED_DATE	CASE_STATUS	CERTIFICATION_BEGIN_DATE	CERTIFICATION_END_DATE	EMPLOYER_NAME
1	H-400-17156-311783	10/2/17	H-2B	17-Aug-17	Certification Expired	15-Nov-17	16-Jan-18	Superior Midway Games, L
2	H-400-17192-987368	10/2/17	H-2B	05-Sep-17	Certification Expired	01-Dec-17	15-Apr-18	Mount Snow Ltd.
3	H-400-17205-983034	10/2/17	H-2B	08-Sep-17	Certification Expired	07-Dec-17	06-Sep-18	International Resorts Mana
4	H-400-17212-334917	10/2/17	H-2B	09-Sep-17	Certification Expired	08-Dec-17	02-Apr-18	Snake River Lodge Hotel In
5	H-400-17215-839949	10/2/17	H-2B	03-Aug-17	Certification Expired	01-Nov-17	15-Jun-18	Bay Fresh Oyster, Inc.
6	H-400-17226-371113	10/2/17	H-2B	14-Aug-17	Certification Expired	29-Oct-17	09-Jun-18	McManus Farms, Inc.
7	H-400-17227-978025	10/2/17	H-2B	18-Aug-17	Certification Expired	01-Nov-17	30-Nov-17	Landscapes Unlimited, LLC
8	H-400-17229-449199	10/2/17	H-2B	17-Aug-17	Withdrawn	01-Nov-17	31-Aug-18	Lonehollow, L.L.C
9	H-400-17244-530865	10/2/17	H-2B	07-Sep-17	Certification Expired	21-Nov-17	30-Apr-18	Handy Andy Snow Remova

The dataset has 59 columns 9490 observations in total, and could be accessed through the following link: https://docs.google.com/spreadsheets/d/1G8PW0MaNWzG_RfSOsHjFOOzTPD-bkhaABispBZzlUmo/edit#gid=1366487551

One of the main reasons for choosing this dataset is that it is good for analysis. The dataset has sufficient volume and is well qualified for creating above 15 tables in 3NF, which gives us a lot of space to restructure the relational database. Meanwhile, we could understand the H2B application information and make corresponding decisions through the dataset. In this way, we could make full use of the dataset, and create a complete H2B application information platform for DOL. DOL can use it to establish certain recruitment and displacement standards in order to protect workers.

DATA INTRODUCTION

The employment information for H-2B visa dataset from the website of United States Department of Labor has been applied to fulfill the goal of this project (link to the original dataset: https://www.foreignlaborcert.doleta.gov/performancedata.cfm).

This file, disclosed by Office of Foreign Labor Certification, contains administrative data of employers' H-2B Applications for Temporary Labor Certification for Non-Agricultural Workers. There are 9490 applications from October 1st, 2017 to September 30th, 2018, and this dataset collects detailed information(59 variables) for each application.(note that not all variables would be used)

The dataset contains basic information about each submitted case, including unique 'case_number' assigned to each application, 'decision_date' on which the last decision was recorded, 'VISA_class' which refers to H-2B in this case, 'submitted_date' on which the application was received, 'case_status' associated with the last decision, and many other related information.

Detailed employer information is recorded in the dataset as well, including 'employer_name', 'employer_address', 'employer_city', 'employer_phone', and other contact information about employer who requests temporary labor certification.

The dataset also collects information about agent or attorney, including 'agent_attorney_name', 'agent_attorney_city', and other information about the agent or attorney filing the H-2B application on behalf of the employer.

Information about job is also included in the dataset, which contains 'job_title', 'SOC_code' and 'SOC_title' as classified by the Standard Occupational Classification (SOC) System, 'full_time_position', 'basic_number_ of_hours' offered each week, 'basic_rate_of_pay', and other job information.

Sample data: Please see appendix 1: Sample Data

NORMALIZATION

The dataset has 59 variables including information about the application status, employers, employees and agents. Therefore, it's necessary to conduct table normalization before we implement analytics procedures. Normalization is a database design technique which organizes tables in a manner that reduces redundancy and dependency of data. It divides larger tables to smaller tables and links them using relationships.

There are three steps for the normalization. First, we need to generate tables in the first normal form (1NF). The requirements for 1NF is that the domains of all table attributes must be atomic and there cannot be repeating attributes. For the "major" column, we noticed that the contents are not atomic, because some rows include two majors within the same cell. Therefore, we separated those majors and created a new table named "major". Besides, there are duplicated values for the addresses of employers, employees and agents. So we created a table "address", and put all the location information such as address, city, state, etc. to the table. In order to accomplish this process, we created three temporary data frames to store unique location data of employers, employees and agents, then we combined the data together, and ultimately the table "address" is populated with unique location data from employers, employees and agents.

Second, we came to the next step of generating tables in the second normal form (2NF). Since there are no composite keys in the tables created in the 1NF, no adjustments are needed for the second step.

Finally, we are in the position to create tables in the third normal form (3NF). The requirements for 3NF is that tables must be in 2NF and every non-key attribute must be non-transitively dependent on the key. Therefore, we separated "job_title" from the main data frame and created a new table named "job". Agent, employer, and employee have multiple addresses and phones, so separate tables are built for them. Besides, jobs in the same industry may have the same soc_code so we create tables for soc_code and job. Also, nasics_code can be related to different employer so nasics table is build. In order to ensure no information is missed, separate tables are created to connect case_number and the attributes mentioned above. We generated 20 tables in total, and they are listed as follows: “address”, “soc_system”, “agents_info”, “employer”, “employer_phone”, “lawfirm”, “cases”, “job”, “case_lawfirm”, “case_phones”, “case_agents”, “case_job”, “address_employers”, “address_agents”, “address_employee”, “job_soc”, “naics”, “case_naics”, “major”, “case_major”.

Code for normalization: Please see appendix 2: Code for Normalization

ER Diagram: Please see the attached pdf

Lucidchart Link for ER Diagram: https://www.lucidchart.com/invitations/accept/380307d1-55f3-4540-97f0-e239a9ffb234

https://www.lucidchart.com/invitations/accept/977540a4-e8fb-424f-8390-29f47f3830d6

ETL PROCESS

With the database and all tables created (3NF), it is now time to extract, transform and load (ETL) the dataset into the database. In order to do so we will have to perform several data transformations on the loaded dataframe, df, in order to create all primary keys and maintain proper relationships.

And our team will explain in detail the reasoning and process of ETL for each table. Within the “job” table, there are duplicated job titles so we cannot simply add a column with incrementing integer numbers for the primary key of “job” as this would lead to multiple primary keys. Therefore, we created a dataframe with unique job titles and added a column with incrementing integer numbers. We also made sure that there are no missing values for this dataframe. Finally, we can populate the “job” table with the value in the dataframe we created.

For the table “address”,we named the columns of the table based on the common location information, and they are “address”, “city”, “state”, “postal_code”, “country”, “province” and “county”. For agents and employees that do not have “address”, “postal_code”, “country” and “province”, we fill them with null values. Unique “address_id” is determined by unique combination of these variables. In order to populate the data into database, a temporary table for address including id and address information is created and we inserted the temporary table to the database.

For “soc_system”, “soc_id” is created by unique combination of “soc_code” and “soc_title”. Temporary table “socs_df” is developed in advance before data is populated to the database.

For “agents_info”, “agent_id” is created based on the “agent_attorney_name”. Similarly, for the table “employer”, “employer_id” is created based on “employer_name” and its business name for submitting labor condition application. Thus, “agents_df” and “employer_df” are created. Then we populated the data to the table “agents_info” and “employer”.

We created the table “employer_phone” because employers might have multiple phone numbers. Then we created a temporary data frame “phone_df” with unique “employer_id” and “employer_phone”. And “phone_id” was added based on “employer_id” and “employer_phone”. Finally, we populated the table “employer_phone” with “phone_df”.

For the table “cases”, we need to process some attributes before they were populated into the database. “basic_rate_of_pay” is converted to numeric by removing $ sign. Besides, NAs are filled for those rows that do not have values for dates and time such as “decision_date”, “submitted_date”, “certification_begin_date”, and “hourly_work_schedule_am”. To populate the “cases” table, we first selected relevant columns from the main data frame to “cases_df”, and then used *dbWriteTable* function populate.

For the “case_phones” table, since we already added “phone_id” to the dataframe “phone_df”, we used “phone_df” as a lookup table so that we can add “phone_id” to the main dataframe. Then we sliced the main data frame to get the table “case_phones”, which includes “phone_id” and “case_number”. Finally, we can populate the “case_phones” table with the value in the data frame we created.

Similarly, since we already added “agent_id” to the dataframe “agents_df”, we used “agents_df” as a lookup table so that we can add “agent_id” to the main dataframe. Then we sliced the main data frame to get the table “case_agents”, which includes “agent_id” and “case_number”. Finally, we can populate the “case_agents” table with the value in the data frame we created.

Just like the table “case_agents”, since we had “soc_id” in the table “socs_df” and “job_id” in the table “job_df”, we used “socs_df” and “socs_df” as lookup tables so that we can add “soc_id” and “job_id” to the main dataframe. Then we sliced the main data frame to get the table “case_job”, which includes “soc_id”, “job_id” and “case_number”.

Within “address_employers” table, there are “address_id” and “employer_id” so that employer information and address information can be linked together. We added “address_id” and “employer_id” to the main data frame, then we sliced the main data frame to get the table “address_employers”, which includes “address_id” and “employer_id”.

For “address_agents” table, there are “address_id” and “agent_id” so that employer information and address information can be linked together. I added “address_id” and “agent_id” to the main dataframe, then we sliced the main data frame to get the table “address_agents”, which includes “address_id” and “agent_id”.

Since job is classified by the Standard Occupational Classification (SOC) System, “soc_code” is associated with the job being requested for temporary labor certification. Thus, we created the “job_soc” table to demonstrate the relationship between “job_titile” and “soc_code”. First, the “job_id” was created for unique job title in the “job” table, and “soc_id” was created for unique soc_code in the “soc_system” table, which were then added to the original table. Next, we retrieved the “job_id” and “soc_id” columns from the main table to create a new data frame, “job_soc_df”, and then stored those data into the “job_soc” table.

“naics” table was created based on unique “naics_code”. First, we created “naics_id” for each unique “naics_code”, and then added “naics_id” in to the original table. Next, we retrieved the “naics_id” and “naics_code” columns from the main table to create a new data frame, “naics_df”. Finally, we stored this data frame in the “naics” table.

Employer is classified by the North American Industrial Classification System (NAICS), and “naics_code” is associated with the employers requesting permanent labor certification. As such, we created the “employer_naics” table to demonstrate the relationship between “employer_id” and “naics_code”. First, the “naics_id” was created in the “naics” table, and “employer_id” was created in the “employer” table, which are both added to the original table. We then retrieved the “naics_id” and “employer_id” from the main table to create a new data frame, “employer_naics_df”, and stored this data frame in the “employer_naics” table.

Since there may be two or more majors related to one case, we created a “major” table to store the information about major. First, we created “major_id” for each unique “major” and then added “major_id” into the original table. Next, we created a new data frame by retrieving the “major_id” and “major” from the main table. Finally, this data frame was stored in the “major” table.

To demonstrate the relationship between “case_number” and “major”, we then created a “case_major” table to store that information. First, “major_id” was created in the “major” table and added to the original table. Next, we retrieved the “case_number” and “major_id” columns from the original table to create a new data frame “case_major_df”. Finally, we stored this data frame in the “case_major” table.

Code for ETL process: Please see appendix 3: Code for ETL Process

Github Link: **https://github.com/mingruihong/SQL-PROJECT-TEAM5**

ANALYTICAL PROCEDURES

After the ETL process, all data is populated to the database so that we can implement analytical procedures and come up with useful analysis for analysts and C-level officers. Our team provide

10 important insights for our client, and we divided them into two sections, one is for our C-executives and the other is for analysts.

ANALYTICAL PROCEDURES FOR C-EXECUTIVES

Since we are hired by the US government to analyze the H2B data and provide insightful conclusions for them to understand the application information and make corresponding decisions in the future, we looked into the data and came up with the following questions to answer:

Q1: What are the top 3 job titles for each state in the US?

This question aims to find out the different types of job markets for the states, for example, some states may provide enormous job opportunities such as sales associates, while others may focus on finding the professionals in landscape industry. From the result, we can see that the the most popular job in Alaska is Jewelry Salesperson, while in Alabama is Landscape Laborer.

	state character varying (30)	job_title character varying (100)	job_id character varying (500)	count bigint	rank bigint
1		Home attendant	1801	1	1
2	AK	Jewelry Salesperson	1171	7	1
3	AK	Sales Associate	997	3	2
4	AK	Salmon Roe Quality Control ...	1531	3	2
5	AL	Landscape Laborer	97	19	1
6	AL	GROUNDS MAINTENANCE ...	544	8	2
7	AL	Oyster Shuckers	34	4	3

```
SELECT *
FROM
(SELECT state,job_title, job_id, count,RANK()over( partition by state order by count desc) as
rank
FROM
(SELECT state,job_title, case_job.job_id, COUNT (case_job.job_id)as count
FROM case_job JOIN job ON case_job.job_id=job.job_id
JOIN address_employers ON address_employers.case_number=case_job.case_number
JOIN address ON address_employers.address_id=address.address_id
GROUP BY state,job_title,case_job.job_id
ORDER BY state,job_title,case_job.job_id , count desc ) as foo) as foo1
WHERE rank<=3;
```

Q2: For different education levels, what are the top 3 job titles in the percentage of all jobs for the specific education level?

This question aims to research the different education levels for different jobs, and what jobs take account of a large percentage for certain education levels. From the result, the most common jobs for master's degree are winemakers and bilingual preschool administrator, and landscape laborers are the most common jobs without education levels.

	education_level character varying (30)	job_title character varying (100)	job_id character varying (500)	count bigint	rank bigint	percentage double precision
32	Master's	Winemaker	284	1	1	0.5
33	Master's	BILINGUAL (SPANISH-ENGL...	1819	1	1	0.5
34	None	Landscape Laborer	97	1914	1	0.213496932515337
35	None	Laborer	141	256	2	0.0285554935861684
36	None	Housekeeper	6	251	3	0.0279977691020636
37	Other Degree (JD, MD, etc.)	TRUCK DRIVER	1240	2	1	0.285714285714286

```
SELECT *
FROM
(SELECT education_level,job_title, job_id, count,RANK()over( partition by
education_level order by count desc) as rank,percentage
FROM
(SELECT education_level,job_title, case_job.job_id, COUNT (case_job.job_id)as
count, (COUNT(case_job.job_id)/ CAST(SUM(COUNT(*)) OVER (PARTITION
BY education_level) AS float) ) as percentage
FROM case_job JOIN cases ON case_job.case_number=cases.case_number
JOIN job ON case_job.job_id=job.job_id
GROUP BY education_level,job_title,case_job.job_id
ORDER BY education_level,job_title,case_job.job_id , count desc ) as foo) as
foo1
WHERE rank<=3;
```

Q3: What are the top 10 job titles grouped by employee worksites?

The purpose of this question is to locate the job markets for job seekers and help the government understand the different number of job offerings from different states and what types of jobs they should pay attention to, considering their numbers. From the result, Texas is the largest job market in the US for H2B workers and landscape laborer is the most popular job.

	job_title character varying (100)	employee_state character varying (30)	count bigint
1	Landscape Laborer	TX	320
2	Landscape Laborer	PA	198
3	Landscape Laborer	OH	140
4	Laborer	TX	134

```
SELECT job_title,state AS employee_state,COUNT(job_title)
FROM address
JOIN address_employee ON address_employee.address_id = address.address_id
```

```
JOIN case_job ON case_job.case_number = address_employee.case_number
JOIN job ON job.job_id = case_job.job_id
GROUP BY job_title,employee_state
ORDER BY COUNT(job_title) DESC
LIMIT 10;
```

Q4: What law firms are efficient in completing approved cases?

This question can be insightful in finding the most efficient law firms when it comes to completing an approved case. Efficiency is very important not only for H2B workers but also for government, because it will speed up the application process, and ultimately lead to government resources optimization. Therefore, government can choose to partner with some law firms to increase working efficiency. Thus public satisfaction will be increased as well. And from the result, the most efficient law firms are Youngblood & Associates and JKJ Workforce Agency.

	certification_begin_date date	certification_end_date date	sum bigint	lawfirm_name character varying (100)
1	2018-10-15	2018-11-01	17	YOUNGBLOOD & ASSOCIAT...
2	2018-10-01	2018-10-20	19	JKJ WORKFORCE AGENCY, ...
3	2018-10-14	2018-11-03	20	JKJ WORKFORCE AGENCY, ...

```
SELECT certification_begin_date, certification_end_date,SUM(certification_end_date-
certification_begin_date),lawfirm_name
FROM cases
JOIN case_lawfirm ON cases.case_number = case_lawfirm.case_number
JOIN lawfirm ON lawfirm.lawfirm_id = case_lawfirm.lawfirm_id
WHERE case_status IN ('Partial Certification', 'Certification')
GROUP BY certification_begin_date, certification_end_date,lawfirm_name
HAVING SUM(certification_end_date-certification_begin_date)>0
ORDER BY SUM(certification_end_date-certification_begin_date);
```

Q5: What is the average working hours for different kinds of jobs?

Since different jobs require different working schedules, the government may want to know the working hours of these jobs. Therefore, C-level officers can understand whether some jobs require extra working time, and determine whether the working hours are reasonable or violate employment law. From the result, most jobs require a 8 - 9 hours' working time, which is reasonable. However, some jobs such as tree trimmers and cooks need more than 10 hours' working time.

	job_title character varying (100)	avg_work_schedule interval
30	Tree Trimmers, Pruners & L...	11:00:00
31	Roofer-helpers	08:00:00
32	FOREST LABORER	09:00:00
33	Pipelayer Helper	09:00:00
34	GARDENER/GROUNDSKEE...	08:00:00

SELECT job_title, avg(hourly_work_schedule_pm - hourly_work_schedule_am) as avg_work_schedule
FROM cases
JOIN case_job ON case_job.case_number = cases.case_number
JOIN job ON job.job_id = case_job.job_id
GROUP BY job_title;

Q6: What are the top 10 employers that offer a large number of job opportunities?

This question is useful for the government to find out companies that offer many job opportunities in the US. These companies are usually large-scale and carry some corporate social responsibilities. Therefore, it's important that the government is aware of these companies, and can even offer some benefits to them if they make great contributions to society. Then, these companies will operate better and recruit more people, which will reduce unemployment rate in the US.

	employer_name character varying (100)	count bigint
1	Imperial Pacific Internation...	46
2	Allagash Maple Products Inc.	37
3	Landscapes Unlimited, LLC	28

SELECT employer_name, COUNT(job_id)
FROM employer
JOIN address_employers ON address_employers.employer_id = employer.employer_id
JOIN case_job ON case_job.case_number = address_employers.case_number
GROUP BY employer_name
ORDER BY COUNT(job_id) DESC
LIMIT 10;

ANALYTICAL PROCEDURES FOR ANALYSTS

It’s crucial for analysts to gain some insights from the H2B data, such as what industries have the most chance of being approved a H2B visa, so that these insights can be used for the analytics team and relevant stakeholders. Besides, with some useful techniques, analysts can predict the possibility of certificating a H2B visa, and find out the common features of those applicants who got a H2B visa. Therefore, our team came up with the following questions:

Q1: What industries have the great chance of being approved a H2B visa?

This question is for analysts to find out the pattern behind the visa application, because there are some industries that are favored by the visa authorization agency, and workers in these industries have a better chance to get approved for their H2B visa. NAICS codes are used by government authorities to differentiate types of business according to their process of production, and the code “561730” has the most number of approved jobs, which refers to the landscaping services.

	naics_code character varying (500)	case_status_count bigint
1	561730	2902
2	721110	647
3	713910	463

SELECT naics_code, COUNT(case_status) AS case_status_count
FROM naics
JOIN case_naics ON naics.naics_id = case_naics.naics_id
JOIN cases ON case_naics.case_number = cases.case_number
WHERE case_status IN ('Partial Certification', 'Certification')
GROUP BY naics_code
ORDER BY case_status_count DESC;

Q2: What are the average hourly wages for different job titles and what jobs offer the highest average hourly wage?

This question aims to tell analysts the wage ranges for different types of jobs so that they can identify what jobs generate high wages. The finding is insightful for the analytics team and they can explore the factors leading to higher wages for different jobs. From the finding, the job “Initial Production Manager” has the highest hourly wage of $71.19.

	job_title character varying (100)	avg numeric
1	BILINGUAL (SPANISH-ENGL...	[null]
2	Initial Production Manager	71.1900000000000000
3	Purchasing Manager	68.8200000000000000
4	Android Developer	65.0300000000000000

SELECT job_title,AVG(basic_rate_of_pay)
FROM cases
JOIN case_job ON cases.case_number=case_job.case_number
JOIN job ON job.job_id=case_job.job_id
WHERE pay_range_unit='Hour'
GROUP BY job_title
ORDER BY AVG(basic_rate_of_pay) DESC;

Q3: How can analysts predict the application status such as certification or withdrawn for future applicants?

Code for this question: Please see appendix 4: Code for Analytical Procedures

By using decision tree models, we aimed to predict the possibility of getting a H2B visa. Decision tree models allow users to develop classification systems that predict or classify future observations based on a set of decision rules. Since there are relationships between the case status and variables such as types of jobs and employees' information, we split the data into train and test data, and put these variables into the model to make predictions. The findings can be very insightful for analysts to indicate the possibility of a case status for future applicants.

	Certification	Certification Expired	Denied	Partial Certification	Partial Certification Expired	REJECTED	Withdrawn
45	0.8341615	0.06708075	0.0000000	0.08322981	0.015527950	0.00000000	0.0000000
53	0.0000000	0.00000000	0.6267281	0.00000000	0.000000000	0.03686636	0.3364055
57	0.0000000	0.00000000	0.7043796	0.00000000	0.000000000	0.06204380	0.2335766
58	0.8966107	0.05383082	0.0000000	0.04500142	0.004557106	0.00000000	0.0000000
61	0.8966107	0.05383082	0.0000000	0.04500142	0.004557106	0.00000000	0.0000000
62	0.8966107	0.05383082	0.0000000	0.04500142	0.004557106	0.00000000	0.0000000

The above chart is a partial of the final result. There are seven possible case statuses for a case, and they are "Certification", "Certification Expired", "Denied", "Partial Certification", "Partial Certification Expired", "Rejected" and "Withdrawn". For the first one, it shows that there is

83.4% possibility that the H2B visa is going to be approved, while the second applicant has 62.7% possibility that the visa is going to be denied.

Q4: What are the top known majors for applicants whose H2B visa got approved?

There are some common features for applicants whose H2B visa got approved and it can be useful for analysts to find out the pattern. Here we analyzed the common majors for applicants whose H2B visa got approved. It can be used for applicants to find out what majors they should study to apply for the visa successfully. From the findings, there are many unknown majors, but the majors Physical Education and Sport & Leisure would be easier to apply for H2B visa.

	major character varying (300)	case_status_count bigint
6	none	23
7	Physical Education, Sport & ...	19
8	na	18
9	N/a	3
10	General Studies	3

```
SELECT major, COUNT(case_status) AS case_status_count
FROM major
JOIN case_major ON major.major_id = case_major.major_id
JOIN cases ON case_major.case_number = cases.case_number
WHERE case_status IN ('Partial Certification', 'Certification')
GROUP BY major
ORDER BY case_status_count DESC;
```

IMPLEMENTATION TOOLS FOR DIFFERENT CUSTOMERS

As for implementation tools, analysts will be granted with authorities to query and analyze the data. It is common that various research projects need different accesses to the H2B dataset. However, in order to ensure the data security and data responsibility, limited authorizations should be assigned to these projects. For example, researchers who focus on unemployment status might only have access to basic certification information such as "case_number," "decision_date", "visa_type", "submitted_date", "case_status", "certification_begin_date", "certification_end_date", "nbr_workers_requested" and "nbr_workers_certified". Labor union could have access to information related to wages and work hours.

In these cases, views will be created as below:

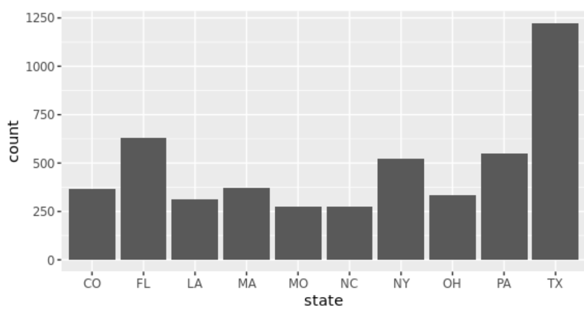
```
CREATE VIEW labor_union AS
SELECT
```

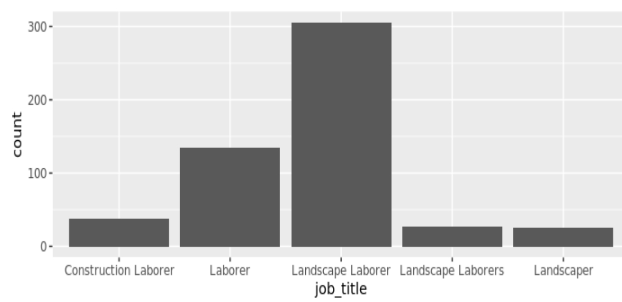
case_number,number_of_hours,hourly_work_schedule_am,hourly_work_schedule_pm,basic_rate_of_pay,overtime_rate_from,overtime_rate_to,pay_range_unit, name, dept_name FROM cases JOIN case_job ON cases.case_number=case_job.case_number;

Then views will be granted to special analysts or roles:

CREATE ROLE union;
GRANT union TO Amit;
GRANT SELECT ON labor_union TO union

Dashboard and graphs can be created by analysts to help c-suite or non-technical personnel to review the results. Specifically, metabase and Python/R can be used to develop the visualization and report for executives. Some of the examples of the visualization by using R are shown below:





(top 10 H2B applications by state) *(the most popular job applications in TX)*

Connecting programming language with database can facilitate programmers who need H2B data to conduct analysis or visualization processes. For example, an analyst can build classification model to predict the possibility of certification status based on the information of applicants. Also, it is convenient that R/Python has packages for visualization such as ggplot2 and matplotlib.

Outside users might need to interact with the database such as visitors to Labor of Department page and applicants who need to submit and review the status of their applications. Private information such as employee’s name, address, and phone might be restricted to them and open data are reported publicly after analysis. Applicants can interact with the database remotely after username and password are verified. They are able to complete forms online and submitted forms through the system and data will be inserted to the database system.

Inner non-technical users apart from analysts and specific executive managers might include auditors who are in charge of certification can change the status of case and update the information through application system. Personnel apart from those who mentioned above should not be granted to adjust the database for data privacy issues.

PLAN FOR REDUNDANCY AND STORAGE

Since our database are divided into 20 tables, it might be inconvenient for analysts to query database because many joins are needed to output the desired result. As a result, sometimes denormalization is needed to facilitate query database and analysis. For example, denormalized tables can be created as we did in our dashboard to visualize features and patterns of different entities such as agents, employers, jobs and cases.

H2B data should be stored in relational database on-premises. Concerning the 3 V's of our data, the growth of volume is about 10.15% in 2018, which is far less than the growth rate of big data. Also, the speed of data processing does not need to to be as high as transactions. As for the variety of data, most of the data are in traditional data format and elements are relatively unchangeable for a long time. Besides, some private information such as phone, address and name are under the issue of security. Labor of department should be responsible for the information completeness and security.

BUSINESS DASHBOARD

Business dashboard is the combination of performance management and business intelligence, which can provide a powerful way to communicate strategy within an organization, and monitor and analyze business performance.

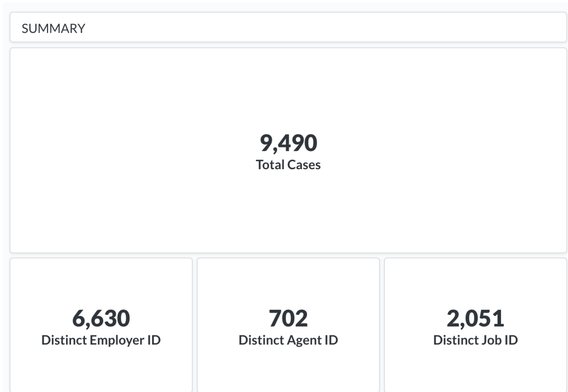
First of all, business dashboards can visualize all valuable information as needed, providing on-demand visibility and insight. Interactive visualizations make business performance and issues easier to observe, serving as an effective solution to the overwhelming amount of data.

Furthermore, business dashboards are continuously iterated when the business grows and changes, giving users ongoing performance measurement capabilities. There is no need for users to wait for monthly report to monitor performance or respond to changing conditions. Reports can be shared quickly between users, allowing the free flow of information between key players. And strategies can be quickly adjusted to fit real-time conditions.

Lastly, business dashboards can provide better understanding of the overall business as well as performance in each functional department. For each group of users, the information and analytical capability is tailored and appropriate to their role.

OVERVIEW

Link: http://s19db.apan5310.com:3105/public/dashboard/d44e67ac-f31f-411f-8e08-1d5764cb6bf4



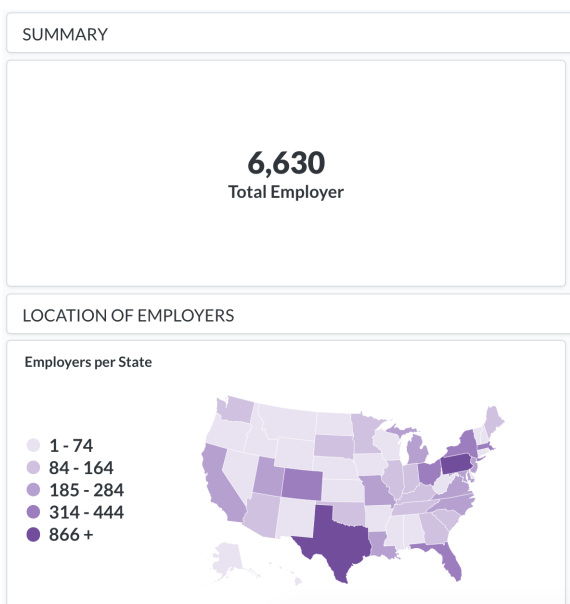
In this database, there are 9490 submitted applications requesting a H2B visa, 6630 employers requesting temporary labor certification, 702 agents filling the application on behalf of the employer, and 2051 agriculture job titles.

AGENT INFORMATION

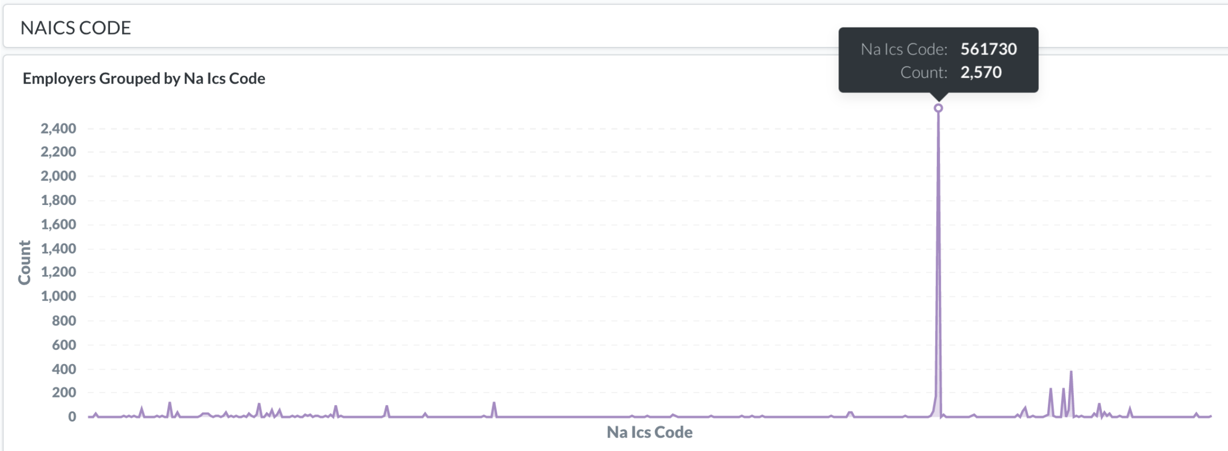
Link: http://s19db.apan5310.com:3105/public/dashboard/957a77c8-bb87-44e8-9bed-53334c062eb3

EMPLOYER INFORMATION

Link: http://s19db.apan5310.com:3105/public/dashboard/ac6d5532-e761-49cd-9cf5-ebba5c5d2564



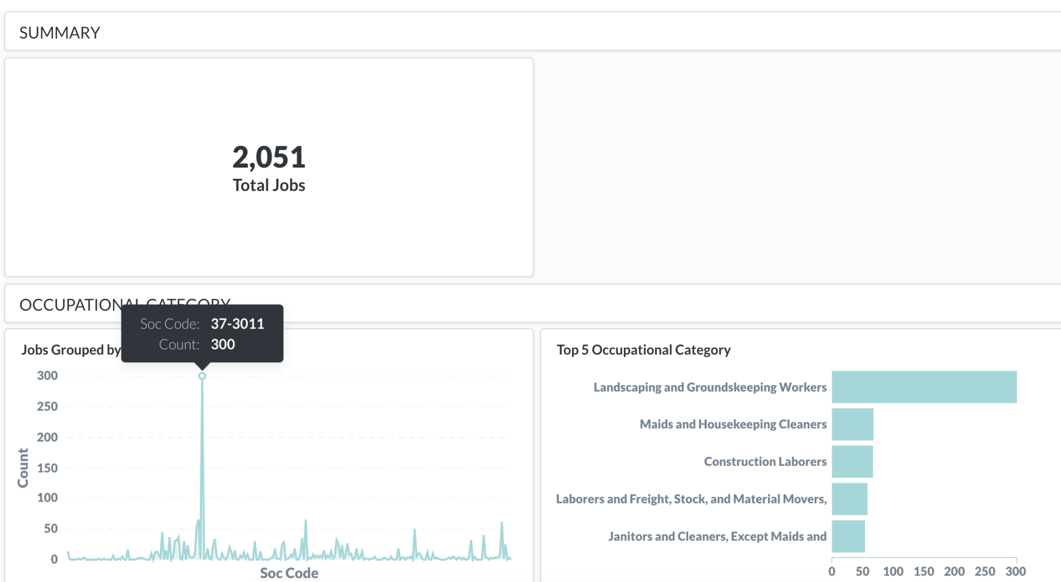
In this database, there are 6630 employers who have applied for temporary labor certification. And most of them are located in Texas State and Pennsylvania State.



Most of the employers are operating in the Landscaping Services Industry, the NAICS Code of which is 561730. This information is in accordance with the job category.

JOB INFORMATION

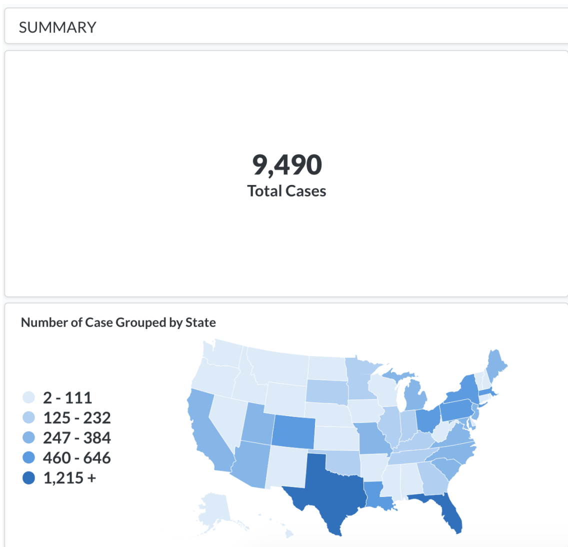
Link: http://s19db.apan5310.com:3105/public/dashboard/b11d7519-e637-4c72-9f48-12edf33ba113



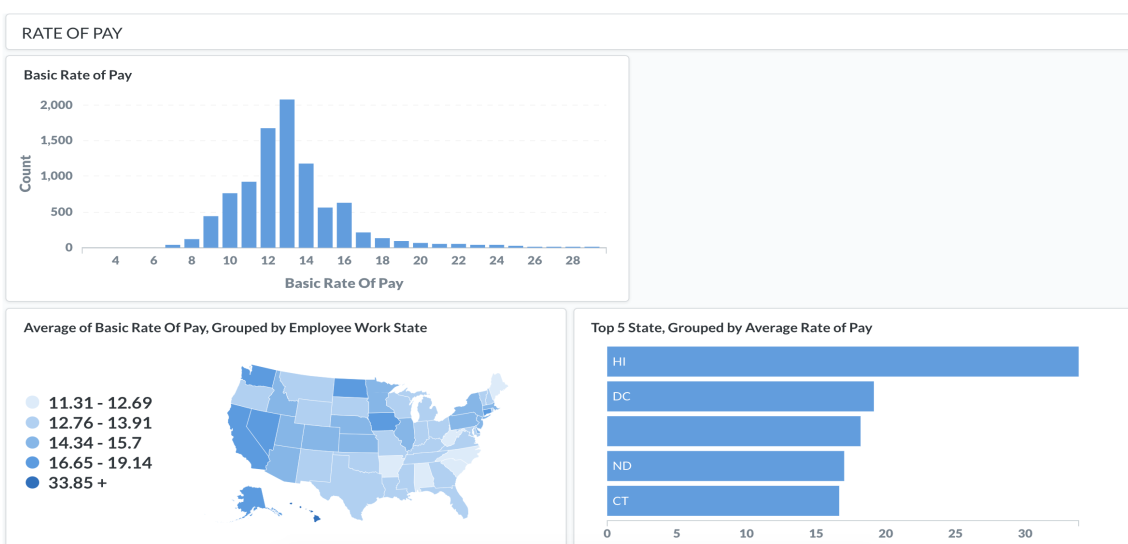
There is a total number of 2051 different jobs in this database. Among those jobs, the most popular category is landscaping and groundskeeping service, the SOC Code of which is 37-3011.

CASE INFORMATION

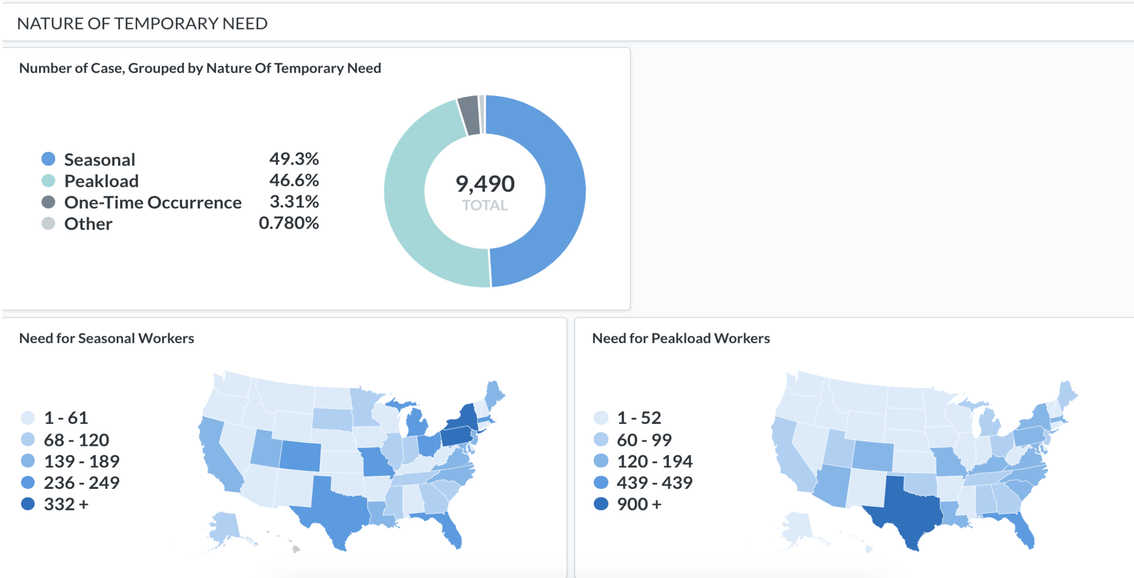
Link: http://s19db.apan5310.com:3105/public/dashboard/f8bc8361-fd64-4bf1-8c7f-62fd351e6535



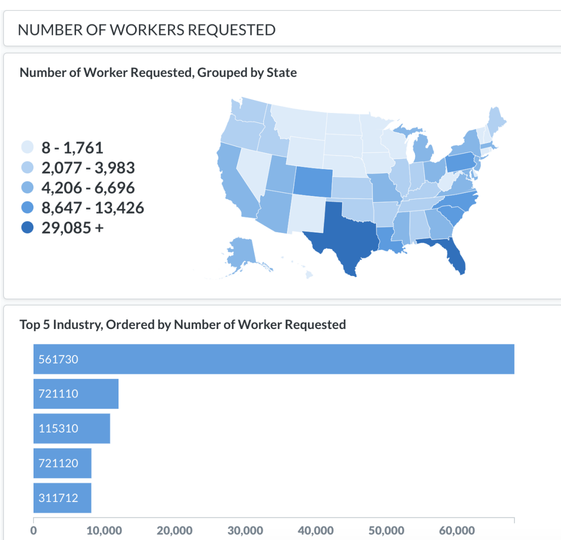
There is a total of 9490 applications submitted for processing to the ETA National Processing Center. And Texas State and Florida State are the most intended area of employment among foreign workers, which is consistent with the distribution of employers.



The basic rate of pay offered by employers range from $7 to $29 per hour, and most of them lie between $12 and $14. Grouped by the employee work state, the average rate of pay is highest in Hawaii State, which is more than $30 per hour. However, only one employer is located in Hawaii State.



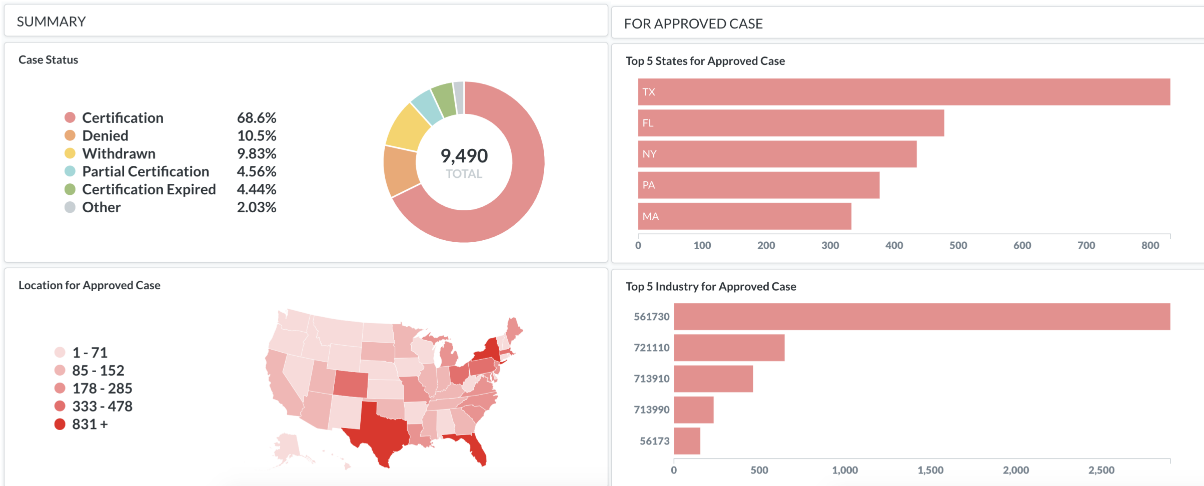
H2B visa nonimmigrant program permits U.S. employers to temporarily hire non-professional foreign workers to work on a seasonal, peak load, or one-time basis. In this case, most of the applications are of seasonal or peak load need. New York State and Pennsylvania State have a strong need for seasonal workers, while Texas State has a strong need for peak load workers.



Texas State and Florida State are requesting more workers than any other states, and most of the workers are requested by Landscaping Services Industry(NAICS Code: 561730). This information is consistent with the employer information.

CASE STATUS INFORMATION

Link: http://s19db.apan5310.com:3105/public/dashboard/cb9bf8da-4c37-4443-a056-b2199a08d067



About 73% of the applications are certified or partial certified, and about 20% of the applications are denied or withdrawn. Texas State, Florida State and New York State have the highest number of approved cases, and the Landscaping Services Industry has the highest number of approved cases. These results is in accordance with the location and distribution of applications.

CONCLUSION

More and more people are choosing to work in America than ever before. In 2018, approximately 9 million non-immigrant visas were issued by issuing office in US, which means there aren't enough U.S. workers who are willing and able to fill the need. H2B visa permits foreign workers to perform temporary non-agricultural services or labor on a one-time, seasonal, peak-load or intermittent basis. And our goal of analyzing this dataset is to evaluate the overall situation of the employment of temporary laborers in U.S. We hope we could provide some valuable insights about the approval of H2B visa and about industries that lack native workers and need to hire foreign labors. We may also further analyze and compare the unemployment conditions in US.

To achieve our goal, we first conduct table normalization which organizes our tables in RDMS format. RDMS, relational database management system, allows multiple database operators to change the database simultaneously without collisions while keeping individual updated records. Also the use of database can be restricted by database administrator which ensures data security and consistency. The speed of RDMS provides users ongoing measurement capabilities which is very important in business development. Then, we extract, transform and load (ETL) the dataset into the database. ETL process simplifies the operating process by using flexible representation so that users could focusing on the ideas and functions. After the ETL process, all data is populated to the database so that we can implement analytical procedures and generate valuable insights for analysts and C-level customers.

For analysts, they will be granted with authorities to query and analyze the data, and limited authorizations will also be assigned in order to ensure the data security and data responsibility. They can then generate valuable insights, such as the chance of being approved a H2B visa and predictions of the application status.

For C-level customers, we will provide them valuable insights to better understand the application information and make corresponding decisions in the future, including the overall approval rate of H2B visa, the job category which have a strong need for temporary labor, the distribution of applications, and the specific employer or industry that is lacking temporary labors. We will implement business dashboard as the interactive tools for our C-level customers, which provides an effective way to communicate strategies and monitor business performance.

APPENDIX 1: SAMPLE DATASET

	job_title	soc_code	soc_title	agent_attorney_name	agent_attorney_city	agent_attorney_state	employer_name
1	Maintenance Helper	49-9098	Helpers--Installation, Maintenance, and Repair	ROBERT PIERCE	ANNAPOLIS	MD	Superior Midway Games, LLC.
2	Maintenance Helper	49-9098	Helpers--Installation, Maintenance, and Repair	MITCHELL ZWAIK	RONKONKOMA	NY	POOLTASTIC POOL WORK INC
3	Maintenance Helper	49-9098	Helpers--Installation, Maintenance, and Repair	ROBERT PIERCE	ANNAPOLIS	MD	Superior Midway Games, LLC.
4	Maintenance Helper	49-9098	Helpers--Installation, Maintenance, and Repair				AQUASAFE POOL MANAGEMENT, INC.
5	Maintenance Helper	49-9098	Helpers--Installation, Maintenance, and Repair	BRENDA DEARMAS RICCI	NEW ORLEANS	LA	Alma Plantation, LLC
6	Maintenance Helper	49-9098	Helpers--Installation, Maintenance, and Repair	MITCHELL ZWAIK	RONKONKOMA	NY	PELICAN POOLS INC.
7	Maintenance Helper	49-9098	Helpers--Installation, Maintenance, and Repair	JACKIE MITCHELL	BATON ROUGE	LA	Raceland Raw Sugar, LLC
8	Maintenance Helper	49-9098	Helpers--Installation, Maintenance, and Repair	JACKIE MITCHELL	BATON ROUGE	LA	M. A. Patout & Son, Ltd.
9	Maintenance Helper	49-9098	Helpers--Installation, Maintenance, and Repair	ROBERT PIERCE	ANNAPOLIS	MD	Ottaway Amusement Company, Inc.
10	Maintenance Helper	49-9098	Helpers--Installation, Maintenance, and Repair	ROBERT PIERCE	ANNAPOLIS	MD	Spectaculars Incorporated

trade_name_dba	employer_address_1	employer_city	employer_state	employer_postal_code	employer_country	employer_province
N/A	mailing: PO Box 238	Stuart	FL	34995	UNITED STATES OF AMERICA	N/A
	182 EAST MONTAUK HIGHWAY	HAMPTON BAYS	NY	11946	UNITED STATES OF AMERICA	
N/A	mailing: PO Box 238	Stuart	FL	34995	UNITED STATES OF AMERICA	N/A
	7466 NEW RIDGE ROAD, SUITE 18	HANOVER	MD	21076	UNITED STATES OF AMERICA	
	4612 Alma Road	Lakeland	LA	70752	UNITED STATES OF AMERICA	
	509 COUNTY ROAD 39	SOUTHAMPTON	NY	11968	UNITED STATES OF AMERICA	
n/a	P. O. Box 159	Raceland	LA	70394	UNITED STATES OF AMERICA	n/a
n/a	3512 J. Patout Burns Rd.	Jeanerette	LA	70544	UNITED STATES OF AMERICA	n/a
	19650 Straight Creek Rd.	Onaga	KS	66521	UNITED STATES OF AMERICA	N/A
N/A	7029 Nundy Avenue	Gibsonton	FL	33534	UNITED STATES OF AMERICA	N/A

naics_code	case_number	decision_date	visa_type	submitted_date	case_status	certification_begin_date	certification_end_date	employer_address_2
713990	H-400-17156-311783	10/2/17	H-2B	17-Aug-17	Certification Expired	15-Nov-17	16-Jan-18	physical: 3350 SW Deggeller Ct, Palm City, FL
238190	H-400-17313-137421	2/20/18	H-2B	01-Jan-18	Certification	01-Apr-18	15-Oct-18	
713990	H-400-18205-648679	9/28/18	H-2B	29-Aug-18	Certification	15-Nov-18	17-Jan-19	physical: 3350 SW Deggeller Ct, Palm City, FL
561790	H-400-17336-984304	1/8/18	H-2B	02-Dec-17	Certification	01-Mar-18	15-Oct-18	
311311	H-400-18180-579917	7/30/18	H-2B	03-Jul-18	Certification	01-Oct-18	29-May-19	
238990	H-400-17320-060691	2/20/18	H-2B	01-Jan-18	Certification	01-Apr-18	01-Nov-18	
311311	H-400-18141-011943	7/30/18	H-2B	03-Jul-18	Certification	01-Oct-18	30-Jun-19	Hwy. 182 and Mill St.
311311	H-400-18141-265312	7/30/18	H-2B	03-Jul-18	Certification	01-Oct-18	31-Jan-19	n/a
713990	H-400-17254-249991	10/18/17	H-2B	11-Sep-17	Certification Expired	26-Nov-17	01-Feb-18	N/A
713990	H-400-18170-515667	8/13/18	H-2B	17-Jul-18	Certification	15-Oct-18	01-Apr-19	N/A

employer_phone	employer_phone_ext	agent_poc_employer_rep_by_agent	lawfirm_name	nbr_workers_requested	nbr_workers_certified	full_time_position
772-215-2223		Y	THE PIERCE LAW FIRM, LLC	12	12	Y
631-287-3500		Y	ZWAIK GILBERT & ASSOCIATES	6	6	Y
772-215-2223		Y	THE PIERCE LAW FIRM, LLC	12	12	Y
301-850-0143	104	N		4	4	Y
225-627-6666		Y	LAW OFFICES OF BRENDA J. DEARMAS RICCI	18	18	Y
631-287-5135		Y	ZWAIK GILBERT & ASSOCIATES	5	5	Y
985-537-3533		Y	FOREIGN LABOR SOLUTIONS, L.L.C.	2	2	Y
337-276-4593		Y	FOREIGN LABOR SOLUTIONS, L.L.C.	1	1	Y
785-213-5778		Y	THE PIERCE LAW FIRM, LLC	4	4	Y
502-553-9530		Y	THE PIERCE LAW FIRM, LLC	20	20	Y

nature_of_temporary_need	number_of_hours	hourly_work_schedule_am	hourly_work_schedule_pm	basic_rate_of_pay	overtime_rate_from	overtime_rate_to	pay_range_unit
Seasonal	35	08:30 AM	05:00 PM	12.39	18.59	NA	Hour
Seasonal	40	08:00 AM	04:00 PM	14.57	21.86	21.86	Hour
Seasonal	35	08:00 AM	05:00 PM	13.15	19.73	NA	Hour
Seasonal	40	08:00 AM	03:00 PM	16.35	NA	NA	Hour
Seasonal	48	07:00 AM	03:00 PM	11.98	17.97	NA	Hour
Seasonal	40	08:00 AM	04:00 PM	14.57	21.86	21.86	Hour
Seasonal	48	07:00 AM	03:00 PM	14.16	21.24	NA	Hour
Seasonal	48	12:00 AM	12:00 PM	14.16	21.24	NA	Hour
Seasonal	35	08:00 AM	05:00 PM	10.78	16.17	NA	Hour
Seasonal	35	08:30 AM	05:00 PM	13.80	20.70	NA	Hour

supervise_other_emp	supervise_how_many	education_level	other_education	major	second_diploma	second_diploma_major	training_required	num_months_training
N	NA	None	N/A	N/A	N		N	NA
N	NA	None			N		N	NA
N	NA	None	N/A	N/A	N		N	NA
N	NA	None			N		N	NA
N	NA	None			N		N	NA
N	NA	None			N		N	NA
N	NA	None	n/a	n/a	N		N	NA
N	NA	None	n/a	n/a	N		N	NA
N	NA	None	N/A	N/A	N		N	NA
N	NA	None	NA		N		N	NA

name_required_training	emp_experience_reqd	emp_exp_num_months	employee_worksite_city	employee_worksite_county	employee_work_state	employee_postal_code
	N	NA	Palm City	Martin	FL	34990
	N	NA	Hampton Bays	Suffolk	NY	11946
	N	NA	Palm City	Martin	FL	34990
	N	NA	Norristown	Montgomery County	PA	19403
	Y	3	Lakeland	Pointe Coupee	LA	70752
	N	NA	Southampton	Suffolk	NY	11968
	Y	3	Raceland	Lafourche Parish	LA	70394
	Y	3	Jeanerette	Iberia Parish	LA	70544
	N	NA	Onaga	Pottawatomie	KS	66521
	Y	3	Gibsonton	Hillsborough	FL	33534

other_worksite_location	swa_name	job_idnumber	job_start_date	job_end_date	x	employer_id	agent_id	job_id
N					NA	1	1	1
Y					NA	2030	333	1
N					NA	1	1	1
Y					NA	1379	7	1
N					NA	5986	395	1
Y					NA	2121	333	1
N					NA	6097	134	1
N					NA	6111	134	1
N					NA	118	1	1
N					NA	6257	1	1

APPENDIX 2: CODE FOR NORMALIZATION

```
CREATE TABLE address (
   address_id  varchar(500),
   address      varchar(500),
   city        varchar(30),
   state       varchar(30),
   postal_code varchar(30),
   country     varchar(30),
   county      varchar(30),
   province       varchar(30),
   PRIMARY KEY (address_id)
 );

CREATE TABLE soc_system (
   soc_id    varchar(500),
   soc_code  varchar(30),
   soc_title  varchar(100),
   PRIMARY KEY (soc_id)
 );

CREATE TABLE agents_info (
   agent_id            varchar(500),
   agent_attorney_name varchar(100),
   PRIMARY KEY (agent_id)
 );

CREATE TABLE employer (
   employer_id   varchar(500),
   employer_name  varchar(100),
   trade_name_dba   varchar(100),
   PRIMARY KEY (employer_id)
 );

CREATE TABLE employer_phone (
   phone_id  varchar(500),
   employer_id   varchar(500),
   employer_phone  varchar(100),
   PRIMARY KEY (phone_id),
```

```
  FOREIGN KEY (employer_id) REFERENCES employer (employer_id)
 );

CREATE TABLE lawfirm (
  lawfirm_id   varchar(500),
  lawfirm_name  varchar(100),
  PRIMARY KEY (lawfirm_id)
 );

CREATE TABLE cases (
  case_number          varchar(30),
  decision_date        date,
  visa_type            varchar(30),
  submitted_date       date,
  case_status          varchar(100),
  certification_begin_date   date,
  certification_end_date   date,
  agent_poc_employer_rep_by_agent   varchar(30),
  nbr_workers_requested    varchar(500),
  nbr_workers_certified    varchar(500),
  full_time_position    varchar(30),
  nature_of_temporary_need   varchar(100),
  number_of_hours    integer,
  hourly_work_schedule_am    time,
  hourly_work_schedule_pm   time,
  basic_rate_of_pay    numeric(10,2),
  overtime_rate_from   numeric(10,2),
  overtime_rate_to    numeric(10,2),
  pay_range_unit   varchar(30),
  supervise_other_emp   varchar(30),
  supervise_how_many   varchar(30),
  education_level    varchar(30),
  other_education    varchar(100),
  second_diploma    varchar(30),
  training_required    varchar(30),
  emp_experience_reqd    varchar(30),
  emp_exp_num_months    varchar(30),
  other_worksite_location    varchar(30),
  PRIMARY KEY (case_number)
```

```
);

CREATE TABLE job (
   job_id varchar(500),
   job_title varchar(100),
   PRIMARY KEY (job_id)
);

CREATE TABLE case_lawfirm (
   case_number varchar (30),
   lawfirm_id varchar(500),
   PRIMARY KEY (case_number, lawfirm_id),
   FOREIGN KEY (case_number) REFERENCES cases (case_number),
   FOREIGN KEY (lawfirm_id) REFERENCES lawfirm (lawfirm_id)
);

CREATE TABLE case_phones (
   case_number varchar (30),
   phone_id varchar(500),
   PRIMARY KEY (case_number, phone_id),
   FOREIGN KEY (case_number) REFERENCES cases (case_number),
   FOREIGN KEY (phone_id) REFERENCES employer_phone (phone_id)
);

CREATE TABLE case_agents (
   case_number varchar (30),
   agent_id    varchar(500),
   PRIMARY KEY (case_number, agent_id),
   FOREIGN KEY (case_number) REFERENCES cases (case_number),
   FOREIGN KEY (agent_id) REFERENCES agents_info (agent_id)
);

CREATE TABLE case_job (
   case_number varchar (30),
   job_id   varchar(500),
   PRIMARY KEY (case_number, job_id),
   FOREIGN KEY (case_number) REFERENCES cases (case_number),
   FOREIGN KEY (job_id) REFERENCES job (job_id)
);
```

```
CREATE TABLE address_employers (
   employer_id    varchar (500),
   address_id   varchar (500),
   case_number  varchar(30),
   PRIMARY KEY (employer_id, address_id, case_number),
   FOREIGN KEY (employer_id) REFERENCES employer (employer_id),
   FOREIGN KEY (address_id) REFERENCES address (address_id),
   FOREIGN KEY (case_number) REFERENCES cases (case_number)
 );

CREATE TABLE address_agents (
   agent_id    varchar (500),
   address_id   varchar (500),
   case_number  varchar (30),
   PRIMARY KEY (agent_id, address_id, case_number),
   FOREIGN KEY (agent_id) REFERENCES agents_info (agent_id),
   FOREIGN KEY (address_id) REFERENCES address (address_id),
   FOREIGN KEY (case_number) REFERENCES cases (case_number)
 );

CREATE TABLE address_employee (
   case_number varchar(30),
   address_id    varchar (500),
   PRIMARY KEY (case_number, address_id),
   FOREIGN KEY (case_number) REFERENCES cases (case_number),
   FOREIGN KEY (address_id) REFERENCES address (address_id)
 );

CREATE TABLE job_soc (
   job_id    varchar (500),
   soc_id   varchar (500),
   PRIMARY KEY (job_id, soc_id),
   FOREIGN KEY (job_id) REFERENCES job (job_id),
   FOREIGN KEY (soc_id) REFERENCES soc_system (soc_id)
 );

CREATE TABLE naics (
   naics_id   varchar (500),
```

```
  naics_code   varchar (500),
  employer_id   varchar (500),
  PRIMARY KEY (naics_id),
  FOREIGN KEY (employer_id) REFERENCES employer (employer_id)
 );

CREATE TABLE case_naics (
  case_number varchar(30),
  naics_id    varchar (500),
  PRIMARY KEY (case_number, naics_id),
  FOREIGN KEY (case_number) REFERENCES cases (case_number),
  FOREIGN KEY (naics_id) REFERENCES naics (naics_id)
 );

CREATE TABLE major (
  major_id   varchar (500),
  major   varchar(300),
  PRIMARY KEY (major_id)
 );

CREATE TABLE case_major (
  case_number varchar(30),
  major_id    varchar (500),
  PRIMARY KEY (case_number, major_id),
  FOREIGN KEY (case_number) REFERENCES cases (case_number),
  FOREIGN KEY (major_id) REFERENCES major (major_id)
 )
```

APPENDIX 3: CODE FOR ETL PROCESS

```
# Import necessary packages
require('RPostgreSQL')

# Load the PostgreSQL driver
drv <- dbDriver('PostgreSQL')

# Create a connection
con <- dbConnect(drv, dbname = 'H2B_visa4',
                 host = 's19db.apan5310.com', port = 50105,
                 user = 'postgres', password = 'y1d8v68m')

# Pass the SQL statements that create all tables
# Please see appendix 2: Code for Normalization

# Execute the statement to create tables
dbGetQuery(con, stmt)

# Load the csv file in a dataframe, df
df <- read.csv('H-2B_Disclosure_Data_FY2018_EOY.csv', stringsAsFactors=FALSE)

# Make dataframe columns lowercase to match PostgreSQL
names(df) <- tolower(names(df))
head(df)

# basic_rate_of_pay have $ sign, convert to numeric
str(df$basic_rate_of_pay)
df$basic_rate_of_pay = as.numeric(gsub("\\$", "", df$basic_rate_of_pay))
head(df$basic_rate_of_pay)

# Replace empty dates and time with NA
df$decision_date[df$decision_date == ''] <- NA
df$submitted_date[df$submitted_date == ''] <- NA
df$certification_begin_date[df$certification_begin_date == ''] <- NA
df$certification_end_date[df$certification_end_date == ''] <- NA
df$hourly_work_schedule_am[df$hourly_work_schedule_am == ''] <- NA
df$hourly_work_schedule_pm[df$hourly_work_schedule_pm == ''] <- NA
```

```
# Replace other NA columns with empty space
df$employer_phone[is.na(df$employer_phone)] <- ''
df$trade_name_dba[is.na(df$trade_name_dba )] <- ''
df$major[is.na(df$major)] <- ''


#Create the "address" dataframe of unique values and add the address_id
df$employer_address <- paste(df$employer_address_1, df$employer_address_2)
df$employer_county=''
empl_colnames <- c('employer_address', 'employer_city', 'employer_state',
                   'employer_postal_code', 'employer_country', 'employer_province','employer_county')
df$employer_province[is.na(df$employer_province)]=''
#df$employer_province[df$employer_province=='N/A']=''
empl_address_df <- df[empl_colnames][!duplicated(df[empl_colnames]),]
colnames(empl_address_df) <- c('address', 'city', 'state', 'postal_code', 'country',
'province','county')

df$agent_attorney_city[is.na(df$agent_attorney_city)]=''
df$agnt_address <- ''
df$agnt_postal_code <- ''
df$agnt_country <- ''
df$agnt_province <- ''
df$agnt_county <- ''
agnt_address_df <- df[c('agnt_address', 'agent_attorney_city', 'agent_attorney_state',
'agnt_postal_code', 'agnt_country',
'agnt_province','agnt_county')][!duplicated(df[c('agnt_address', 'agent_attorney_city',
'agent_attorney_state', 'agnt_postal_code', 'agnt_country', 'agnt_province','agnt_county')]),]
colnames(agnt_address_df) <- c('address', 'city', 'state', 'postal_code', 'country',
'province','county')
address_df <- rbind(empl_address_df, agnt_address_df)

df$employee_country <- ''
df$employee_province <- ''
df$employee_address<-''
employee_address_df <-
df[c('employee_address','employee_worksite_city','employee_work_state','employee_postal_cod
e','employee_country','employee_province','employee_worksite_county')][!duplicated(df[c('empl
oyee_address','employee_worksite_city','employee_work_state','employee_postal_code','employ
ee_country','employee_province','employee_worksite_county')]),]
```

```
colnames(employee_address_df) <- c('address', 'city', 'state', 'postal_code', 'country',
'province','county')
address_df <- rbind(address_df, employee_address_df)
address_df=address_df[c('address', 'city', 'state', 'postal_code', 'country',
'province','county')][!duplicated(address_df[c('address', 'city', 'state', 'postal_code', 'country',
'province','county')]),]
address_df$address_id=1:nrow(address_df)

# Push the "address" dataframe to database
dbWriteTable(con, name='address', value=address_df, row.names=FALSE, append=TRUE)

# Create the "soc_system" dataframe of unique soc codes and names and add the soc_id
socs_df <- df[c('soc_code', 'soc_title')][!duplicated(df[c('soc_code', 'soc_title')]),]
socs_df$soc_id <- 1:nrow(socs_df)

# Push the "soc_system" dataframe to database
dbWriteTable(con, name='soc_system', value=socs_df, row.names=FALSE, append=TRUE)

# Create the "agents_info" dataframe of unique agent names and add the agent_id
agents_df <- data.frame('agent_attorney_name' = unique(df$agent_attorney_name))
agents_df$agent_id <- 1:nrow(agents_df)
agents_df <- agents_df[complete.cases(agents_df), ]

# Push the "agents_info" dataframe to database
dbWriteTable(con, name='agents_info', value=agents_df, row.names=FALSE, append=TRUE)

# Create the "employer" dataframe of unique employer names and and trade_name_dba add the
employer_id
employer_df <- df[c('employer_name','trade_name_dba')][!duplicated(df[c('employer_name',
'trade_name_dba')]),]
employer_df$employer_id <- 1:nrow(employer_df)

# Push the "employer" dataframe to database
dbWriteTable(con, name='employer', value=employer_df, row.names=FALSE, append=TRUE)

# Add the "employer_id" to the main dataframe, df, using employer_df as a lookup table
df=merge(df,employer_df,by.x = c('employer_name', 'trade_name_dba'),by.y =
c('employer_name', 'trade_name_dba'),no.dups = TRUE,sort=FALSE,all.x=T)
```

```
employer_id_list <- apply(df[c('employer_name', 'trade_name_dba')], 1, function(x) {
employer_df$employer_id[(employer_df$employer_name == x[1]) &
(employer_df$trade_name_dba == x[2])] })
df$employer_id <- employer_id_list

# Create the "employer_phone" dataframe of unique employer_phone and add the phone_id and
employer_id
phone_df <- df[c('employer_phone','employer_id')][!duplicated(df[c('employer_phone',
'employer_id')]),]
phone_df$phone_id <- 1:nrow(phone_df)

# Push the "employer_phone" dataframe to database
dbWriteTable(con, name='employer_phone', value=phone_df, row.names=FALSE,
append=TRUE)

# Create the "lawfirm" dataframe of unique lawfirm names and add the lawfirm_id
lawfirm_df <- data.frame('lawfirm_name' = unique(df$lawfirm_name))
lawfirm_df$lawfirm_id <- 1:nrow(lawfirm_df)
lawfirm_df <- lawfirm_df[complete.cases(lawfirm_df), ]

# Push the "lawfirm" dataframe to database
dbWriteTable(con, name='lawfirm', value=lawfirm_df, row.names=FALSE, append=TRUE)

# Create the "cases" dataframe
cases_df <- df[c('case_number', 'decision_date', 'visa_type', 'submitted_date',
                 'case_status', 'certification_begin_date', 'certification_end_date',
                 'agent_poc_employer_rep_by_agent', 'nbr_workers_requested',
                'nbr_workers_certified', 'full_time_position', 'nature_of_temporary_need',
                'number_of_hours', 'hourly_work_schedule_am', 'hourly_work_schedule_pm',
                'basic_rate_of_pay', 'overtime_rate_from', 'overtime_rate_to', 'pay_range_unit',
                'supervise_other_emp', 'supervise_how_many', 'education_level', 'other_education',
                'second_diploma', 'training_required', 'emp_experience_reqd', 'emp_exp_num_months',
                'other_worksite_location')]

# Push the "cases" dataframe to database
dbWriteTable(con, name='cases', value=cases_df, row.names=FALSE, append=TRUE)

# Create the "job" dataframe of unique job title and add the job_id
job_df <- data.frame('job_title' = unique(df$job_title))
```

```
job_df$job_id <- 1:nrow(job_df)
job_df <- job_df[complete.cases(job_df), ]

# Push the "job" dataframe to database
dbWriteTable(con, name='job', value=job_df, row.names=FALSE, append=TRUE)

# Add the "lawfirm_id" to the main dataframe, df, using lawfirm_df as a lookup table
lawfirm_id_list <- sapply(df$lawfirm_name, function(x)
lawfirm_df$lawfirm_id[lawfirm_df$lawfirm_name == x])
df$lawfirm_id <- lawfirm_id_list

# Slice the main dataframe to get the "case_lawfirm" table
case_lawfirm_df <- df[c('case_number', 'lawfirm_id')][!duplicated(df[c('case_number',
'lawfirm_id')]),]

# Push the "cases_agents" dataframe to database
dbWriteTable(con, name='case_lawfirm', value=case_lawfirm_df, row.names=FALSE,
append=TRUE)

#Add the "phone_id" to the main dataframe, df, using socs_df as a lookup table
df=merge(df,phone_df,by.x = c('employer_phone', 'employer_id'),by.y = c('employer_phone',
'employer_id'),no.dups = TRUE,sort=FALSE,all.x=T)

# Slice the main dataframe to get the "case_phones" table
case_phones_df <- df[c('case_number', 'phone_id')][!duplicated(df[c('case_number',
'phone_id')]),]

# Push the "case_phones" dataframe to database
dbWriteTable(con, name='case_phones', value=case_phones_df, row.names=FALSE,
append=TRUE)

#Add the "agent_id" to the main dataframe, df, using agents_df as a lookup table
agent_id_list <- sapply(df$agent_attorney_name, function(x)
agents_df$agent_id[agents_df$agent_attorney_name == x])
df$agent_id <- agent_id_list

# Slice the main dataframe to get the "case_agents" table
case_agents_df <- df[c('case_number', 'agent_id')][!duplicated(df[c('case_number', 'agent_id')]),]
```

 # Push the "cases_agents" dataframe to database
dbWriteTable(con, name='case_agents', value=case_agents_df, row.names=FALSE, append=TRUE)

 # Add the "job_id" to the main dataframe, df, using job_df as a lookup table
job_id_list <- sapply(df$job_title, function(x) job_df$job_id[job_df$job_title == x])
df$job_id <- job_id_list

 # Slice the main dataframe to get the "case_job" table
case_job_df <- df[c('case_number', 'job_id')][!duplicated(df[c('case_number', 'job_id')]),]

 # Push the "case_job" dataframe to database
dbWriteTable(con, name='case_job', value=case_job_df, row.names=FALSE, append=TRUE)

#Add the "address_df" to the main dataframe
df=merge(df,address_df,by.x = c('employer_address', 'employer_city', 'employer_state', 'employer_postal_code', 'employer_country', 'employer_province','employer_county'),by.y = c('address','city','state','postal_code','country','province','county'),no.dups = TRUE,sort=FALSE,all.x=T)
colnames(df)[76] <- "employer_address_id"

 # Add the "address_employer_df" to the main dataframe
address_employer_df <-df[c('employer_id', 'employer_address_id','case_number')][!duplicated(df[c('employer_id', 'employer_address_id','case_number')]),]
colnames(address_employer_df) <- c('employer_id','address_id','case_number')

 # Push the "address_employer" dataframe to database
dbWriteTable(con, name='address_employers', value=address_employer_df, row.names=FALSE, append=TRUE)

 # Add the "address_df" to the main dataframe
df=merge(df,address_df,by.x = c('agnt_address', 'agent_attorney_city', 'agent_attorney_state', 'agnt_postal_code', 'agnt_country', 'agnt_province','agnt_county'),by.y = c('address','city','state','postal_code','country','province','county'),no.dups = TRUE,sort=FALSE,all.x=T)
colnames(df)[77] <- "address_id"

 # Add the "address_agents_df" to the main dataframe

```
address_agents_df <- df[c('agent_id', 'address_id','case_number')][!duplicated(df[c('agent_id',
'address_id','case_number')]),]
colnames(address_agents_df) <- c('agent_id','address_id','case_number')

dbWriteTable(con, name='address_agents', value=address_agents_df, row.names=FALSE,
append=TRUE)

#Add the "address_df" to the main dataframe
df=merge(df,address_df,by.x =
c('employee_address','employee_worksite_city','employee_work_state','employee_postal_code','e
mployee_country','employee_province','employee_worksite_county'),by.y =
c('address','city','state','postal_code','country','province','county'),no.dups = TRUE,sort=FALSE)
colnames(df)[78] <- "address_id"

# Add the "address_employee_df" to the main dataframe
address_employee_df <-df[c('case_number', 'address_id')][!duplicated(df[c('case_number',
'address_id')]),]
colnames(address_employee_df) <- c('case_number','address_id')

dbWriteTable(con, name='address_employee', value=address_employee_df, row.names=FALSE,
append=TRUE)

# Add the "soc_id" to the main dataframe, df, using socs_df as a lookup table
soc_id_list <- apply(df[c('soc_code', 'soc_title')], 1, function(x) {
socs_df$soc_id[(socs_df$soc_code == x[1]) & (socs_df$soc_title == x[2])] })
df$soc_id <- soc_id_list

#Slice the main dataframe to get the "job_soc" table
job_soc_df <- df[c('job_id', 'soc_id')][!duplicated(df[c('job_id', 'soc_id')]),]

# Push the "job_soc" dataframe to database
dbWriteTable(con, name='job_soc', value=job_soc_df, row.names=FALSE, append=TRUE)

# create 'naics' dataframe of unique code and add the naics_id
naics_df <- df[c('naics_code', 'employer_id')][!duplicated(df[c('naics_code', 'employer_id')]),]
naics_df$naics_id <- 1:nrow(naics_df)

# Push the "naics" dataframe to database
dbWriteTable(con, name='naics', value=naics_df, row.names=FALSE, append=TRUE)
```

```
#Add the "naics_id" to the main dataframe
df<- merge(df,naics_df,by.x = c('naics_code','employer_id'),by.y =
c('naics_code','employer_id'),no.dups = TRUE,sort=FALSE)

#Slice the main dataframe to get the "case_naics" table
case_naics_df <- df[c('case_number','naics_id')][!duplicated(df[c('case_number','naics_id')]),]

#Push the "case_naics" dataframe to database
dbWriteTable(con, name='case_naics', value=case_naics_df, row.names=FALSE,
append=TRUE)

# Create the "major" dataframe of unique major and add the major_id
major_df <- data.frame('major' = unique(df$major))
major_df$major_id <- 1:nrow(major_df)
major_df <- major_df[complete.cases(major_df), ]

# Push the "major" dataframe to database
dbWriteTable(con, name='major', value=major_df, row.names=FALSE, append=TRUE)

#Add the "major_id" to the main dataframe, df, using agents_df as a lookup table
major_id_list <- sapply(df$major, function(x) major_df$major_id[major_df$major == x])
df$major_id <- major_id_list

# Slice the main dataframe to get the "case_major" table
case_major_df <- df[c('case_number', 'major_id')][!duplicated(df[c('case_number', 'major_id')]),]

# Push the "case_major" dataframe to database
dbWriteTable(con, name='case_major', value=case_major_df, row.names=FALSE,
append=TRUE)
```

APPENDIX 4: CODE FOR ANALYTICAL PROCEDURES

ANALYTICAL PROCEDURES FOR ANALYSTS

Q3

```
stmt4<- "SELECT *
        FROM cases  NATURAL JOIN case_lawfirm
        NATURAL JOIN case_agents
        NATURAL JOIN case_job
        NATURAL JOIN case_naics
        NATURAL JOIN case_major  "
df4=dbGetQuery(con, stmt4)
df4=df4[,-c(1:4,6:7,14:15,21,23)]
df4$agent_poc_employer_rep_by_agent[df4$agent_poc_employer_rep_by_agent=='']='Y'
df4$full_time_position[df4$full_time_position=='']='Y'
df4$nature_of_temporary_need[df4$nature_of_temporary_need=='']='Seasonal'
df4$supervise_other_emp[df4$supervise_other_emp=='']='N'
df4$education_level[df4$education_level=='']='None'
df4$second_diploma[df4$second_diploma=='']='N'
df4$training_required[df4$training_required=='']='N'
df4$emp_experience_reqd[df4$emp_experience_reqd=='']='N'
df4$other_worksite_location[df4$other_worksite_location=='']='Y'
df4$pay_range_unit[df4$pay_range_unit=='']='Hour'
df4$case_status=as.factor(df4$case_status)
df4$agent_poc_employer_rep_by_agent=as.factor(df4$agent_poc_employer_rep_by_agent)
df4$nbr_workers_requested=as.numeric(df4$nbr_workers_requested)
df4$nbr_workers_certified=as.numeric(df4$nbr_workers_certified)
df4$full_time_position=as.factor(df4$full_time_position)
df4$nature_of_temporary_need=as.factor(df4$nature_of_temporary_need)
df4$pay_range_unit=as.factor(df4$pay_range_unit)
df4$supervise_other_emp=as.factor(df4$supervise_other_emp)
df4$education_level=as.factor(df4$education_level)
df4$second_diploma=as.factor(df4$second_diploma)
df4$training_required=as.factor(df4$training_required)
df4$emp_experience_reqd=as.factor(df4$emp_experience_reqd)
df4$emp_exp_num_months=as.numeric(df4$emp_exp_num_months)
df4$other_worksite_location=as.factor(df4$other_worksite_location)
df4$job_id=as.numeric(df4$job_id)
df4$major_id=as.numeric(df4$major_id)
```

```
df4$lawfirm_id=as.numeric(df4$lawfirm_id)
df4$agent_id=as.numeric(df4$agent_id)
df4$naics_id=as.numeric(df4$naics_id)
df4$nbr_workers_requested[is.na(df4$nbr_workers_requested)]=20.63
df4$nbr_workers_certified[is.na(df4$nbr_workers_certified)]=19.89
df4$number_of_hours[is.na(df4$number_of_hours)]=38.76
df4$basic_rate_of_pay[is.na(df4$basic_rate_of_pay)]=19.61
df4$overtime_rate_from[is.na(df4$overtime_rate_from)]=32.8
df4$overtime_rate_to[is.na(df4$overtime_rate_to)]=20.36
df4$emp_exp_num_months[is.na(df4$emp_exp_num_months)]=0
library(caTools)
set.seed(100)
split = sample.split(df4,SplitRatio = 0.7)
train = df4[split,]
test = df4[!split,]
library(rpart)
tree = rpart(case_status~.,data=train)
predTree = predict(tree,newdata=test)
```