DTSC 691 Project Proposal Template

*Great Basin Tree Company*

By: Matt Dalgetty

# **Background**

Tree companies are customer-facing service organizations. They often have a small team of managers and sales representatives, and a large number of laborers which actually perform the work. In these organizations there is a disconnect between management and workers. The work is performed in different locations while management remains in the office. This can create negative outcomes for customers and decrease the overall quality of work performed. Another problem these companies face is technological illiteracy. Due to the blue-collar nature of the industry, often both management and workforce are not technologically skilled. The workers will have no ability to interact with the data necessary to make “best practice” work decisions. And management will end up performing repetitive IT tasks which could be easily automated.

This disconnect between management and workers is a problem seen in many industries. What is unique to the arboriculture industry is that workers are performing treatments on living organisms which are essential to the environment, trees. Many of the treatments performed on trees can only safely be done every few years. And some of the trees worked on are very old with complex histories, and quite valuable both to the customers and to the environment for the services they provide. Therefore it is of the utmost importance that the job not only gets done, but maintains the integrity of the tree it is being performed on. It is essential that workers and management have access to the history of work that has been performed on individual trees. In order to accomplish this I will be creating a database that employees at *Great Basin Tree Company* will be able to easily interact with both in the office and on the job site.

For this project I will act as an IT professional and include myself in the hierarchy of a fictional small tree company. I intend to solve this problem of worker access to data. I will create a database that helps keep track of inventory, customers, and trees that have been worked on. Because of the fact that many treatments can only happen once every few years, it is essential that these treatments and their associated trees be included in a database. It is also essential that both managers and employees can easily view these observations before starting, and during, work. That is why this database will be hosted as a webapp to allow users to easily access the database of trees that have been worked on, as well as current supplies necessary to perform various treatments.

# **Problem Objective**

The final project will include a number of outcomes:

1. Project PDF including:
   1. ER Diagram for database
   2. Relational database schema diagram
   3. Implementation of database and creation of data to fill it
   4. Example of SQL database manipulation and queries
2. Interactive webapp of database hosted online
3. Video walkthrough of project

In order to achieve this a variety of software will be used to create diagrams, create a database, interact with the database, and host it as a webapp. These steps will be completed in the order described above which creates a procedural workflow. For example, the high level ER diagram must be completed first in order to create the relational database schema diagram.

The project has two audiences which will engage with the different deliverables. The first audience are the graders of this project at Eastern University. The final project PDF and video walkthrough will be delivered for them. The audience of the webapp includes the graders but is ultimately intended for the fictional workers of *Great Basin Tree Company*. The UI/UX of the app will reflect this. The webapp must be easy to use rd a demographic that may have very little experience with technology.

Some constraints of the project are, limited access to real data (fictional company, data), and the 7 week time frame for delivery. The data problem will be solved by only including a few entries for each table. This will not affect the overall quality of the project as the structure of the database and interactivity of the webapp are the most important parts. Once created it will be very easy to backfill with data if it were to be made available. Because of the time constraints there are

interesting additions to the project that will not be able to be accomplished. In the future it would be interesting to create an interpretation of the database and app for Spanish speaking users, as well as using an AI chatbot to further enhance user experience.

# **Data Description**

**Entities, attributes, and data types**

The attached diagram describes 8 tables, their relationships, and their associated entities. The data used to fill these will be created for the sake of this project and will be entirely fictional. The structure created in this project could be backfilled with actual data from a tree company. The tables and their attributes are as follows:

1. **Employee**
   1. Employee ID INT PK
   2. Phone Number VARCHAR(20)
   3. Address VARCHAR(100)
   4. City VARCHAR(50)
   5. State VARCHAR(50)
   6. Employee Name VARCHAR(100)
2. **Appointment**
   1. Appointment ID INT PK
   2. Work Performed VARCHAR(200) FK
   3. Date DATE
   4. Sales Price DECIMAL(10,2)
   5. Employee ID INT FK
   6. Customer ID INT FK
3. **Customer**
   1. Customer ID INT PK
   2. Phone Number VARCHAR(20)
   3. Customer Name VARCHAR(100)
4. **Property**
   1. Customer ID INT FK
   2. Address VARCHAR(100)
   3. City VARCHAR(50)
   4. State VARCHAR(50)
   5. Region Code VARCHAR(10) FK
5. **Region**
   1. Region Code VARCHAR(10) PK
   2. Distance From Office DECIMAL(10,2)
6. **Tree**
   1. Tree ID INT PK
   2. Health VARCHAR(20)
   3. Estimated Age INT
   4. Work Performed VARCHAR(200)
   5. Date DATE
   6. Tree Species VARCHAR(100)
7. **Sales**
   1. Sales Price DECIMAL(10,2)
   2. Customer ID INT FK
   3. Employee ID INT FK
   4. Appointment ID INT FK
   5. Product ID INT FK
8. **Inventory**
   1. Product ID INT PK
   2. Product Cost DECIMAL(10,2)
   3. Quantity DECIMAL(10,2)
   4. Company Name VARCHAR(100)
   5. SKU VARCHAR(50)
   6. Product Name VARCHAR(100)

**Views, Functions, Trigger, and Stored Procedure**

The project will also include 2 views, 2 functions, 1 trigger, and 1 stored procedure. In the views I will explain how different members of the organization might use them. For the functions, trigger, and procedure, I will explain their purpose and function.

**2 Views**

CREATE VIEW [Dying Trees] AS

SELECT Tree\_ID, Health

FROM Tree

WHERE Health = ‘Poor’;

CREATE VIEW [Low Inventory] AS

SELECT Product\_ID, Quantity

FROM Inventory

WHERE Quantity <= 1;

These views would be used by both managers and laborers. The first view could be used by a sales representative to prepare cold calling customers to see about tree removals.

**2 Functions**

CREATE FUNCTION GetTreeByID(

@TreeID INT

)

RETURNS TABLE

AS

RETURN

(

SELECT \*

FROM Tree

WHERE TreeID = @TreeID

);

CREATE FUNCTION CalculateAverageSales()

RETURNS TABLE

AS

RETURN

(

SELECT \*

e.EmployeeID,

e.EmployeeName,

AVG(s.SalesPrice) AS AverageSales

FROM

Employee e

INNER JOIN

Appointment a ON e.EmployeeID = a.EmployeeID

INNER JOIN

Sales s ON a.AppointmentID = s.AppointmentID

GROUP BY

e.EmployeeID,

e.EmployeeName

);

These functions would be used by managers in the office to create lists of certain tree species in order to understand what kinds of work might be required from them. The managers could also calculate the average sales of an employee to create bonuses.

**1 Trigger**

CREATE TRIGGER UpdateAverageSales

ON Sales

After INSERT

AS

BEGIN

UPDATE Employee

SET AverageSales = (

SELECT AVG(SalesPrice)

FROM Sales

WHERE EmployeeID = inserted.EmployeeID

)

FROM Employee

INNER JOIN inserted ON Employee.EmployeeID = inserted.EmployeeID

END;

This trigger is used to calculate the average sales for an employee after a new record has been inserted into the sales table

**1 Stored Procedure**

CREATE PROCEDURE InsertSalesRecord

@salesPrice DECIMAL(10, 2),

@employeeID INT,

@appointmentID INT,

@customerID INT,

@productID INT

AS

BEGIN

INSERT INTO Sales (SalesPrice, EmployeeID, AppointmentID, CustomerID, ProductID)

VALUES (@salesPrice, @employeeID, @appointmentID, @customerID, @productID)

END;

This stored procedure could improve performance of the database and reduce network traffic as well as be used to create permissions so that only some employees can update these tables

# **Software**

The ER model and Relational Database Schema Diagram will be completed using the lucidchart webapp. The proposal, final project write up, and presentation will be created using the Google suite and placed in the Google drive. The project will also be hosted on Github. The SQL database will be created using PostgreSQL. The database will be interacted with using the psycopg2 package in Python. The coding environment will be a Jupyter notebook. The webapp portion of the project will be created using Flask and hosted on AWS

The webapp functionality will include Create, Read, Update, and Delete data operations. There will be a tree page which will allow workers to create, view, and update records in the Tree table. There will also be an appointment page which will allow workers to View their appointments for the day.

# **Analysis plan**

**Deliverables:**

* Project Proposal including ER Diagram
* Final Project Report containing:
  + Jupyter Notebook
* Webapp
* Project Walkthrough Video

| **Week** | **Activity Details** |
| --- | --- |
| 1 | * Planning and constructing ER diagram. Finishing and submitting proposal for review |
| 2 | * Review critiques on proposal and amend as necessary. Start Git repo and begin creating Relational Database Schema Diagram |
| 3 | * Complete Diagram and create and fill SQL database |
| 4-5 | * Connect database to Python using packages in a Jupyter notebook * Design webapp and host to the web using AWS |
| 6-7 | * Create Project walkthrough Video and Final Project Report * Submit Project |

## **Week goals**

### Week 1

* Plan out ER diagram and begin work on the project proposal template. Complete **ER diagram**. Submit Project Proposal for review by mentor and project coordinator.

### Week 2

* Create a Git repository for the project and include the completed ER diagram. Review critiques on the project proposal and make updates. Create a repo for the project on GitHub. Clone repo in Postgres and start creating tables. Start Relational Database Schema Diagram

### Week 3

* Complete **Relational Database Schema Diagram.** Place into Final Project Report and complete **Table of Contents** and **Data and Functional Requirements** section.
* Finish creating the database in PostgreSQL. Create data and insert it into the database. Place **SQL DDL** statements and views into Final Project Report.

### Week 4 & 5

* Start **Jupyter notebook** and connect to SQL database using Psycopg2 Python package. Construct DML statements and Queries to manipulate the data. Place **DML and Queries** into Final Project Report.
* Design a webapp to read data and call on stored procedures using Plotly Dash. Host webapp to AWS. Add completed **Jupyter Notebook** to GitHub repository

### Week 6 & 7

* Clean up GitHub repository
* Create Final Project Walkthrough Video
* Finish Final Project Report
* Submit Project

# **Presentation plan**

The final Project will be hosted in a Github repository as well as a Google drive accessible by moderators of DTSC-691. The webapp will be accessible through AWS. A final project walkthrough will be available in the Google Drive folder. The walkthrough will begin with an introduction to the project and follow with a demonstration of the database, Jupyter Notebook, and AWS. Finally it will include a demonstration of using the webapp and final thoughts on the project. The webapp will also be available for live access.