WEB-SCRAPER – MAINTENANCE PROCESS

1.0 OVERVIEW AND SCOPE

This document outlines the creation and maintenance of the Web-Scraper project. Information here will allow a greater understanding on how to provide maintenance to the Web-Scraper Tableau workbook file, and to the Python 3.6 scripts used to scrape the necessary data. The user performing maintenance will be expected to know the basics of Tableau, Python, and database management. The Tableau workbook file will be used to create and publish Tableau dashboard visuals to either a Tableau Online server account, or to a Tableau Public server account.

- **Tableau Workbook:** The Tableau workbook contains all worksheets and dashboards necessary to publish to the server.
- **Python Data Collection:** Using the Python scripts, the user will collect the data and insert it into the relevant databases. This data is made available through either a live connection or an extract of the AWS Collector database (whose setup and maintenance is out of this document's scope). The data is displayed in Tableau by abstracting SQL queries.
- **Database SQL creates and queries:** Using SQL Server and SQL, multiple tables were created to store the data scraped by Python. No foreign keys exist between the two tables, so the datetime posted of each table's row is used to make an inner join in Tableau.

2.0 COMPONENTS AND SCOPE

Maintenance may impact the following components:

1. **Tableau Data Refresh Rate:** If the user decides to publish to Tableau Public, new data will not be made available. If a live connection is made by publishing to Tableau Online and securely embedding a password in the dashboard. The end user will then be able to refresh the data at will.
2. **Database Login Credentials:** Connecting to the data source requires AWS wobey-db2 login credentials. When publishing to Tableau server, the password will be embedded into the workbook so validated users will not require their own publishing/editing credentials. The same holds true for Tableau Public.
3. **Python Scripts:** Passing certain arguments to the Python scripts will allow a proper connection to:
    - scrape data,
    - login and insert into a database,
    - and send email alerts when inserts fail.

3.0 PROCEDURE COMPONENTS: TABLEAU WORKBOOK

Below are the components added to the Web-Scraper Tableau dashboard:

3.1 **Name Associations**

Database column names have been modified in the following ways:

- **Wind-** [Measure]
  - Anything over 200mph is assumed an error, and displays 0.
- **Weather Condition** [Measure]
  - Because there are many phrases for weather conditions, some have been grouped to ensure easier visual parsing of relevant user data:

```
IF [Phrase] == "Mostly Cloudy"
    THEN "Cloudy"
ELSEIF [Phrase] == "Partly Cloudy"
    THEN "Cloudy (Partly)"
ELSEIF [Phrase] == "Mostly Sunny"
    THEN "Sunny (Mostly)"
ELSE
    [Phrase]
END
```

- **MDY(Month, Day, Year of Datetime Posted)** [Filter]
  - Takes the Reddit post's datetime and converts it into individual days for use in scrolling through available days.
- **Title** [Marks]
  - Each title is sorted in ascending order based on the temperature.
- **Temp** [Marks]
  - Temperature's color filter marks titles with one of six colors (color-temperature scale is displayed to the user).

3.2 **Table Joins** [Data Source]

Below is the only inner joins necessary to blend the two available Collector tables, Posts and Weather. The Data Source panel will require the user to either make custom SQL queries, or use Tableau's UI to specify these inner-joins in Fig. 1:

Table Name	Table Column	Table Column	Table Name
Posts	Datetime Posted	[see below 1.1]	Weather
Posts	Datetime Posted	[see below: 1.2]	Weather

Fig. 1 Table showing the necessary inner-joins to connect four tables.

Below are the two calculations used in joining the two table's in Fig. 1. Because the weather.com site updates on average every ten minutes, the join looks for the first time that is within +/- five minutes of the Reddit post:

- `1.1: DATEADD('minute', 5, [datetime posted (Weather)])`
- `1.2: DATEADD('minute', -5, [datetime posted (Weather)])`

4.0 PROCEDURE COMPONENTS: PYTHON DATA COLLECTION

Below are the essential components of the Web-Scraper Python 3.6 scripts:

4.1 **reddit_collector.py**

Once executed in a command line, the Reddit collector will scrape all www.reddit.com/r/Seattle/new posts. Below are the variables that determine how it gets

the HTML, scrapes it for relevant posts, inserts the posts into the Collector.Posts table, and emails the user if the insert fails.

- Execute the script with the following information:
  `python3 reddit_collector.py https://reddit.com/r/Seattle/new wobey-db2.cepbeby3o59m.us-west-2.rds.amazonaws.com smtp.comcast.net:465`

  The user will then be prompted for a valid email password (for `smtp.comcast.net:465`), and a valid database password (for `wobey-db2.cepbeby3o59m.us-west-2.rds.amazonaws.com`).
- **Requests** is used to get the HTML
- **BeautifulSoup** and **re** are used to initially parse the HTML.
- **Pyodbc** is used to make select and insert queries to the database.
- **Smtplib** is used to send email.
- Reddit's HTML design may change overtime which will require a maintainer to visually parse the HTML, and determine where the relevant tag information for each part of a post is located.
- This type of scraping technically violates Reddit's TOS. Although, it doesn't exceed the number of request to the website, it does send false headers to avoid getting flagged.
  - A **jitter** was added for how long to sleep between requests. This also aids in avoiding detection.
  - This project's ethical repercussions were thoroughly reviewed. As this is only for educational gains, ensuring I don't violate trespass of chattels was paramount. Essentially, my scripts and actions will bring no harm to the website, no malicious intent is made, and my scripts will not inhibit traffic from reaching the website.
- If an insert attempt returns an error, and email will be sent out indicating the post that triggered the error. This will halt the program and require manual aid.

4.2 weather_collector.py

Once executed in a command line, the Weather collector will scrape all www.weather.com current weather for a specific location (i.e. Seattle). Below are the variables that determine how it gets the HTML, scrapes it for relevant weather, inserts the weather into the Collector.Weather table, and emails the user if the insert fails.

- Execute the script with the following information:

  ```
  python3 weather_collector.py
  https://weather.com/weather/today/l/USWA0395:1:US wobey-db2.cepbeby3o59m.us-west-2.rds.amazonaws.com smtp.comcast.net:465
  ```

  The user will then be prompted for a valid email password (for `smtp.comcast.net:465`), and a valid database password (for `wobey-db2.cepbeby3o59m.us-west-2.rds.amazonaws.com`).
- **Requests** is used to get the HTML
- **BeautifulSoup** and **re** are used to initially parse the HTML.

- **Pyodbc** is used to make select and insert queries to the database.
- **Smtplib** is used to send email.
- Weather.com's HTML design may change overtime which will require a maintainer to visually parse the HTML, and determine where the relevant tag information for a weather record.
- This type of scraping technically violates Weather.com's TOS. Although, it doesn't exceed the number of request to the website, it does send false headers to avoid getting flagged.
    - A **jitter** was added for how long to sleep between requests. This also aids in avoiding detection.
    - This project's ethical repercussions were thoroughly reviewed. As this is only for educational gains, ensuring I don't violate trespass of chattels was paramount. Essentially, my scripts and actions will bring no harm to the website, no malicious intent is made, and my scripts will not inhibit traffic from reaching the website.
- If an insert attempt returns an error, and email will be sent out indicating the post that triggered the error. This will halt the program and require manual aid.

4.3 **post.py**

The post.py script only contain the class Post and Posts which help to store multiple posts collected from a single scan of Reddit's page (~25 posts each scrape) when reddict_collector.py is running.

5.0 PROCEDURE COMPONENTS: DATABASE SQL SERVER CREATES AND QUERIES

Below are the SQL Server table creates and queries:

5.1 **Create the table Posts:**

```
CREATE TABLE Posts
(
  id              INT IDENTITY
    PRIMARY KEY,
  datetime_added  DATETIME     NOT NULL,
  username        VARCHAR(20),
  url_path        VARCHAR(MAX),
  title           VARCHAR(300) NOT NULL,
  datetime_posted DATETIME     NOT NULL
)
GO

CREATE UNIQUE INDEX Post_id_uindex
  ON Posts (id)
GO

CREATE UNIQUE INDEX Posts_datetime_posted_uindex
  ON Posts (datetime_posted)
GO
```

5.2 Create the table Weather:

```
CREATE TABLE Weather
(
  id               INT IDENTITY
    PRIMARY KEY,
  datetime_added   DATETIME NOT NULL,
  temp             INT      NOT NULL,
  wind             INT,
  phrase           VARCHAR(30),
  datetime_posted DATETIME NOT NULL
)
GO

CREATE UNIQUE INDEX Weather_column_1_uindex
  ON Weather (id)
GO

CREATE UNIQUE INDEX Weather_datetime_posted_uindex
  ON Weather (datetime_posted)
GO
```

5.3 Check if Weather table already contains the scraped weather (requires pyodbc):

```
sql_exists = textwrap.dedent("""SELECT * FROM Collector.guest.Weather
WHERE datetime_posted = (?);""")
```

5.4 Insert into Weather table (requires pyodbc):

```
sql_insert = textwrap.dedent("""INSERT INTO
Collector.guest.Weather(datetime_added, datetime_posted, temp, wind,
phrase) VALUES (?, ?, ?, ?, ?);""")
```

5.5 Check if Posts table already contains the scraped post (requires pyodbc):

```
sql_exists = textwrap.dedent("""SELECT * FROM Collector.guest.Posts WHERE
datetime_posted = (?) and username = (?);""")
```

5.6 Insert into Posts table (requires pyodbc):

```
sql_insert = textwrap.dedent("""INSERT INTO
Collector.guest.Weather(datetime_added, datetime_posted, temp, wind,
phrase) VALUES (?, ?, ?, ?, ?);""")
```