{

"cells": [

{

"cell\_type": "code",

"execution\_count": 71,

"metadata": {},

"outputs": [],

"source": [

"%matplotlib inline\n",

"import matplotlib.pyplot as plt"

]

},

{

"cell\_type": "code",

"execution\_count": 72,

"metadata": {},

"outputs": [],

"source": [

"import pandas as pd\n",

"import datetime as dt"

]

},

{

"cell\_type": "code",

"execution\_count": 73,

"metadata": {},

"outputs": [],

"source": [

"#Python SQL toolkit and Object Relational Mapper\n",

"import sqlalchemy\n",

"from sqlalchemy.ext.automap import automap\_base\n",

"from sqlalchemy.orm import Session\n",

"from sqlalchemy import create\_engine, func, MetaData, Table"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"#create engine to hawaii.sqlite\n",

"engine = create\_engine (\"sqlite:///C://Users/edwar/Downloads/hawaii.sqlite\")"

]

},

{

"cell\_type": "code",

"execution\_count": 75,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"['id', 'station', 'date', 'prcp', 'tobs']"

]

},

"execution\_count": 75,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"#reflect an existing database into a new model\n",

"metadata = MetaData ()\n",

"\n",

"station = Table (\"station\", metadata, autoload\_with = engine)\n",

"[c.name for c in station.columns]\n",

"[\"id\", \"station\", \"name\", \"latitude\", \"longitude\", \"elevation\"]\n",

"\n",

"measurement = Table (\"measurement\", metadata, autoload\_with = engine)\n",

"[c.name for c in measurement.columns]\n",

"[\"id\", \"station\", \"date\", \"prcp\", \"tobs\"]"

]

},

{

"cell\_type": "code",

"execution\_count": 76,

"metadata": {},

"outputs": [],

"source": [

"#reflect the tables\n",

"Base = automap\_base ()\n",

"Base.prepare (autoload\_with = engine)"

]

},

{

"cell\_type": "code",

"execution\_count": 77,

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"<class 'sqlalchemy.ext.automap.measurement'>\n",

"<class 'sqlalchemy.ext.automap.station'>\n"

]

}

],

"source": [

"#View all of the classes that automap found\n",

"for mapped\_class in Base.classes:\n",

" print (mapped\_class)"

]

},

{

"cell\_type": "code",

"execution\_count": 78,

"metadata": {},

"outputs": [],

"source": [

"#Save references to each table\n",

"Station = Base.classes.station\n",

"Measurement = Base.classes.measurement"

]

},

{

"cell\_type": "code",

"execution\_count": 79,

"metadata": {},

"outputs": [],

"source": [

"#Create our session (link) from Python to the DB\n",

"session = Session (engine)"

]

},

{

"cell\_type": "code",

"execution\_count": 80,

"metadata": {},

"outputs": [],

"source": [

"#Find the most recent date in the data set\n",

"most\_recent\_date = session.query (func.max (Measurement.date)).scalar ()"

]

},

{

"cell\_type": "code",

"execution\_count": 81,

"metadata": {},

"outputs": [],

"source": [

"#Design a query to retrieve the last 12 months of precipitation data.\n",

"#And plot the results. Starting from the most recent dtaa point in the database.\n",

"#Calculate the date one year from the last date in the dataset.\n",

"one\_year\_previous = dt.datetime.strptime (most\_recent\_date, \"%Y-%m-%d\") - dt.timedelta (days = 365)\n",

"one\_year\_previous\_string = one\_year\_previous.strftime (\"%Y-%m-%d\")"

]

},

{

"cell\_type": "code",

"execution\_count": 82,

"metadata": {},

"outputs": [],

"source": [

"#Perform a query to retrieve the date and precipitation scores\n",

"precipitation\_data = session.query (Measurement.date, Measurement.prcp).filter (Measurement.date >= one\_year\_previous).all ()"

]

},

{

"cell\_type": "code",

"execution\_count": 83,

"metadata": {},

"outputs": [],

"source": [

"#Save the query results as a Pandas DataFrame. Explicitly set the column names\n",

"precipitation\_df = pd.DataFrame (precipitation\_data, columns = [\"Date\", \"Precipitation\"])"

]

},

{

"cell\_type": "code",

"execution\_count": 84,

"metadata": {},

"outputs": [],

"source": [

"#Sort the dataframe by date\n",

"precipitation\_df = precipitation\_df.sort\_values (\"Date\")"

]

},

{

"cell\_type": "code",

"execution\_count": 85,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "",

"text/plain": [

"<Figure size 640x480 with 1 Axes>"

]

},

"metadata": {},

"output\_type": "display\_data"

}

],

"source": [

"#Use Pandas plotting with MatPlotLib to plot the data\n",

"precipitation\_df.plot (x = \"Date\", y = \"Precipitation\", \n",

"title = \"Precipitation Over the Last Twelve Months\", rot = 90)\n",

"plt.xlabel (\"Date\")\n",

"plt.ylabel (\"Inches\")\n",

"plt.show ()"

]

},

{

"cell\_type": "code",

"execution\_count": 86,

"metadata": {},

"outputs": [],

"source": [

"#Use Pandas to calculate the summary statistics for the precipitation data\n",

"#Calculate summary statistics for the precipiation data\n",

"summary\_statistics = precipitation\_df [\"Precipitation\"].describe ()"

]

},

{

"cell\_type": "code",

"execution\_count": 87,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Precipitation</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>count</th>\n",

" <td>2015.000000</td>\n",

" </tr>\n",

" <tr>\n",

" <th>mean</th>\n",

" <td>0.176462</td>\n",

" </tr>\n",

" <tr>\n",

" <th>std</th>\n",

" <td>0.460288</td>\n",

" </tr>\n",

" <tr>\n",

" <th>min</th>\n",

" <td>0.000000</td>\n",

" </tr>\n",

" <tr>\n",

" <th>25%</th>\n",

" <td>0.000000</td>\n",

" </tr>\n",

" <tr>\n",

" <th>50%</th>\n",

" <td>0.020000</td>\n",

" </tr>\n",

" <tr>\n",

" <th>75%</th>\n",

" <td>0.130000</td>\n",

" </tr>\n",

" <tr>\n",

" <th>max</th>\n",

" <td>6.700000</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Precipitation\n",

"count 2015.000000\n",

"mean 0.176462\n",

"std 0.460288\n",

"min 0.000000\n",

"25% 0.000000\n",

"50% 0.020000\n",

"75% 0.130000\n",

"max 6.700000"

]

},

"execution\_count": 87,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"#Display the summary statistics as a dataframe\n",

"summary\_statistics\_df = pd.DataFrame (summary\_statistics)\n",

"summary\_statistics\_df"

]

},

{

"cell\_type": "code",

"execution\_count": 88,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"[(9,)]"

]

},

"execution\_count": 88,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"#Design a query to calculate the total number of stations in the dataset\n",

"total\_stations = session.query (func.count (Station.station)).all ()\n",

"total\_stations"

]

},

{

"cell\_type": "code",

"execution\_count": 89,

"metadata": {},

"outputs": [],

"source": [

"#Design a query to find the most active stations (i.e, which stations have yhe most rows?)\n",

"#List the stations and their counts in descending order\n",

"active\_stations = session.query (Measurement.station,\n",

"func.count (Measurement.station)).group\_by (Measurement.station).order\_by (func.count (Measurement.station).desc ()).all ()"

]

},

{

"cell\_type": "code",

"execution\_count": 90,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"[('USC00519281', 2772),\n",

" ('USC00519397', 2724),\n",

" ('USC00513117', 2709),\n",

" ('USC00519523', 2669),\n",

" ('USC00516128', 2612),\n",

" ('USC00514830', 2202),\n",

" ('USC00511918', 1979),\n",

" ('USC00517948', 1372),\n",

" ('USC00518838', 511)]"

]

},

"execution\_count": 90,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"#Disply the result\n",

"active\_stations"

]

},

{

"cell\_type": "code",

"execution\_count": 91,

"metadata": {},

"outputs": [],

"source": [

"#Using the most active station id from the previous query,\n",

"#calculate the lowest, highest and average temperature\n",

"most\_active\_station = active\_stations [0][0]"

]

},

{

"cell\_type": "code",

"execution\_count": 92,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"[(54.0, 85.0, 71.66378066378067)]"

]

},

"execution\_count": 92,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"#Calculate the lowest, highest and average temperature for the most active station\n",

"temperature\_statistics = session.query (func.min (Measurement.tobs).label (\"Lowest Temperature\"),\n",

" func.max (Measurement.tobs).label (\"Highest Temperature\"),\n",

" func.avg (Measurement.tobs).label (\"Average Temperature\")).filter (Measurement.station\n",

" == most\_active\_station).all ()\n",

"temperature\_statistics"

]

},

{

"cell\_type": "code",

"execution\_count": 93,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "",

"text/plain": [

"<Figure size 640x480 with 1 Axes>"

]

},

"metadata": {},

"output\_type": "display\_data"

}

],

"source": [

"#Using the most active station id\n",

"#Query the last 12 months of temperature observation data for this station and\n",

"#plot the results as a histogram\n",

"temperature\_data = session.query (Measurement.date, Measurement.tobs).filter (Measurement.station\n",

"== most\_active\_station, Measurement.date >= one\_year\_previous\_string).all ()\n",

"\n",

"temperature\_df = pd.DataFrame (temperature\_data, columns = [\"Date\", \"Temperature\"])\n",

"\n",

"plt.hist (temperature\_df [\"Temperature\"], bins = 12, label = \"tobs\")\n",

"plt.xlabel (\"Temperature\")\n",

"plt.ylabel (\"Frequency\")\n",

"plt.title (f\"Temperature Observation Data for Station {most\_active\_station} Last Twelve Months\")\n",

"plt.legend ()\n",

"plt.show ()"

]

},

{

"cell\_type": "code",

"execution\_count": 94,

"metadata": {},

"outputs": [],

"source": [

"#Close session\n",

"session.close ()"

]

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.11.9"

}

},

"nbformat": 4,

"nbformat\_minor": 2

}