Overview

The solution aims to extract statistics from two datasets and derive additional metrics. The two datasets, Sensors and Counts, contain data on the pedestrian counting system in a particular area. The solution will load and stage the data in an appropriate format to enable efficient querying.

Business Requirements

The following business requirements are given for the solution:

1. Show the top 10 (most pedestrians) locations by day.
2. Show the top 10 (most pedestrians) locations by month.
3. Determine which location has shown the most decline due to lockdowns in the last two years.
4. Determine which location has had the most growth in the last year.
5. Load and stage data in an appropriate format for future querying.
6. Suggest other metrics that can be derived from these datasets.

Data Model

The data model consists of two tables:

Sensors Table

- sensor_id: the unique identifier for a sensor (primary key)
- sensor_description: description of the sensor
- sensor_name: name of the sensor
- installation_date: the date the sensor was installed
- sensor_status: status of the sensor
- note: additional notes for the sensor
- direction_1: the first direction the sensor is facing
- direction_2: the second direction the sensor is facing
- latitude: the latitude of the sensor
- longitude: the longitude of the sensor
- location: the location of the sensor

Counts Table

- ID: the unique identifier for a count (primary key)
- Date_Time: the date and time of the count
- Year: the year of the count
- Month: the month of the count
- Date: the day of the count
- Day: the day of the week of the count
- Time: the time of the count
- Hourly_Counts: the number of pedestrians counted in an hour
- Sensor_ID: the identifier for the sensor that recorded the count (foreign key)
- Sensor_Name: the name of the sensor that recorded the count

Extracted Metrics

The solution will extract the following metrics from the datasets:

Requested:

1. Top 10 (most pedestrians) locations by day.
2. Top 10 (most pedestrians) locations by month.
3. The location has shown the most decline due to lockdowns in the last two years.
4. The location that has shown the most growth in the last year.

Suggested:

5. Average hourly pedestrian counts by location, day of the week, and month.
6. Median hourly pedestrian counts by location.
7. Total pedestrian counts by day of the week and month.
8. Percent change in pedestrian counts from one year to the next by location.
9. Busiest hour of the day by location.
10. Average hourly pedestrian counts by sensor direction.

Implementation Details

The solution will use Python and SQLite to extract the metrics from the datasets. The data will be staged in SQLite tables for efficient querying. The data will be loaded from CSV files into pandas DataFrames, and then merged based on the sensor_id and Sensor_ID columns. The datetime column will be converted to a datetime object to enable extracting the year, month, day, and hour. The following libraries will be used:

- sqlite3: To connect to the SQLite database and execute SQL queries.
- pandas: To load and manipulate the data.
- os: To set the working directory to the directory containing the script.

Detailed Description of the Method

1. **Requirements gathering and analysis**: We need to identify the requirements and analyse them to understand the data needed to meet those requirements. Based on the requirements provided, we need to extract statistics related to the most crowded locations, the locations that have shown the most decline or growth, and loading and staging data in a format that can be queried in the future. We also need to suggest other metrics that can be derived from the data sets.
2**. Data sources**: We have two data sets with the schema for the Sensors and Counts tables. We will use these tables to extract the required data.
3. Data model: We need to create a data model that represents the data in the datasets and shows the relationships between the tables. We can use this model to create a database and develop queries to extract the required metrics.
4. Diagram: We can create a diagram to visualize the data model and the relationships between the tables.
5. Database creation: We can use the data model to create a database that we can use to develop queries to extract the required metrics.
6. Query development: We can develop queries to extract the required metrics from the database.

We need to extract statistics related to the most crowded locations, the locations that have shown the most decline or growth, and loading and staging data in a format that can be queried in the future. We also need to suggest other metrics that can be derived from the datasets.
In summary:

1. Expected outputs: Top 10 (most pedestrians) locations by day, Top 10 (most pedestrians) locations by month, Location with the most decline due to lockdowns in the last 2 years, Location with the most growth in the last year)
2. Load and stage data in an appropriate format for future querying, and visually mock-up data to represent tables and columns.
3. Identify other metrics that can be derived from the given data sets.

4. Describe the data model and include a diagram.

Based on the provided schema, it appears that the data is stored in two tables: Sensors and Counts. The Sensors table contains information about the sensors used to count pedestrians, and the Counts table contains the actual pedestrian count data.

First, we need to start with a clear understanding of the business requirements and what the data sets contain.

1. Expected Outputs Top 10 (most pedestrians) locations by day: To generate this report, we need to aggregate the pedestrian count data by location and day, sort by the count, and return the top 10 locations. This report will help identify the busiest locations for each day, which could help in scheduling and allocating resources.

2. Top 10 (most pedestrians) locations by month: This report is like the previous one, except we aggregate the data by location and month. It helps to identify the busiest locations for each month, which could be used for planning and analysis.

3. Location with the most decline due to lockdowns in the last two years: To generate this report, we need to compare the pedestrian count data for the past two years and identify the locations that have shown the most decline. This report will help identify the locations that may need more attention and support to recover from the effects of the lockdown.

4. Location with the most growth in the last year: This report is like the previous one, except we need to identify the locations that have shown the most growth in the past year. It helps to identify the locations that are growing faster than others, which could be used for planning and analysis.

To stage the data for future querying, we need to load the data from the provided CSV files into a database. The database should be designed to support querying and analysis. We can use SQLite or any other database that supports SQL queries.

Here is an example SQL query to create the Sensors table:

```
CREATE TABLE Sensors (
  sensor_id INTEGER PRIMARY KEY,
  sensor_description TEXT,
  sensor_name TEXT,
  installation_date DATE,
  sensor_status TEXT,
  note TEXT,
  direction_1 TEXT,
```

```
  direction_2 TEXT,
  latitude FLOAT,
  longitude FLOAT,
  location TEXT
);
```

Here is an example SQL query to create the Counts table:

```
CREATE TABLE Counts (
  ID INTEGER PRIMARY KEY,
  Date_Time DATETIME,
  Year INTEGER,
  Month INTEGER,
  Date INTEGER,
  Day TEXT,
  Time TEXT,
  Hourly_Counts INTEGER,
  Sensor_ID INTEGER REFERENCES Sensors(sensor_id),
  Sensor_Name TEXT
);
```

Once the tables are created, we can load the data into them using an ETL (extract, transform, load) process. We can use Python or any other programming language to write the ETL process. To visually mock-up data to represent tables and columns, we can create a sample data set using a spreadsheet program or a programming language like Python. we can first mock up the tables for the two given datasets:

Table: Sensors

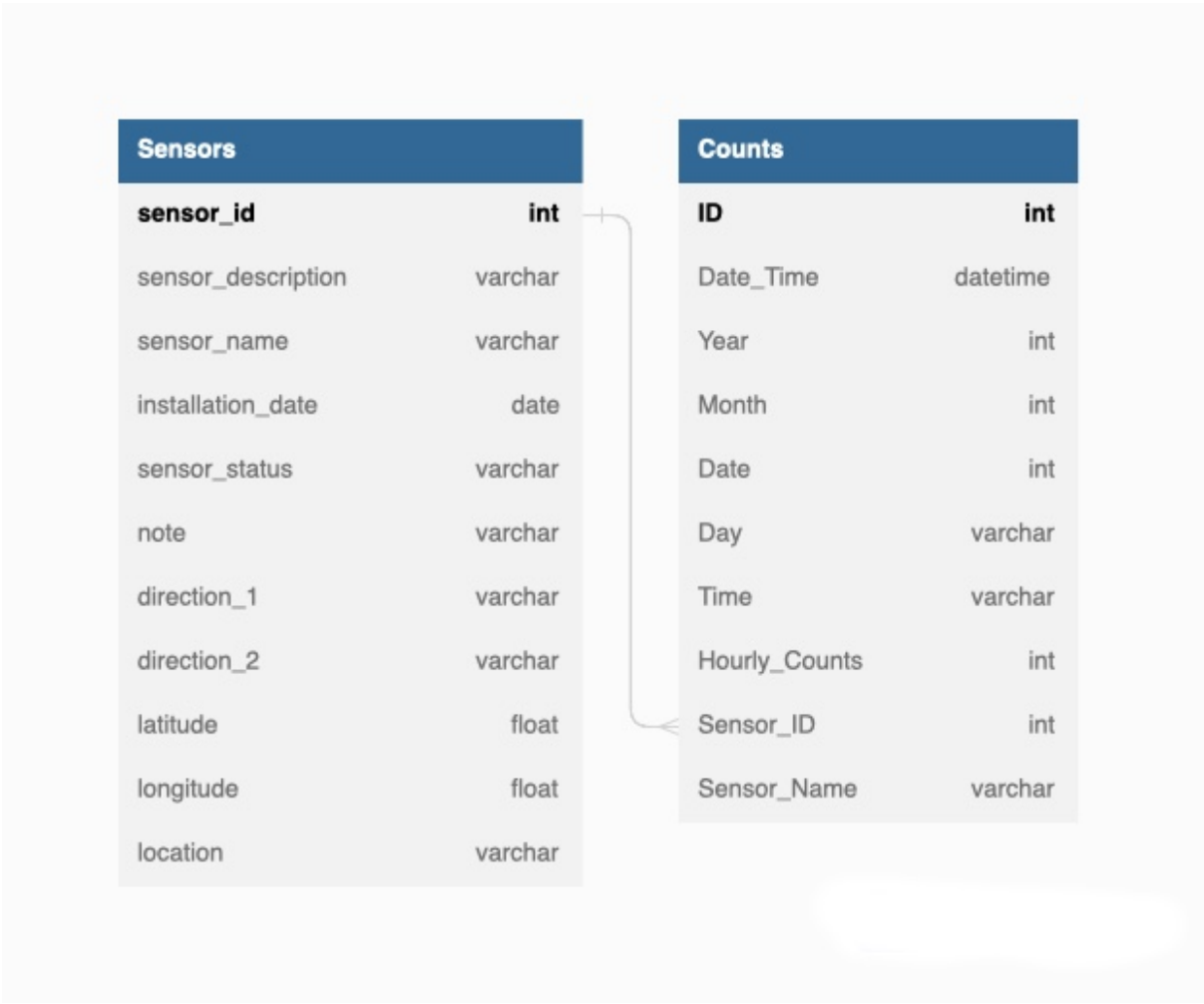
Column Name	Data Type	Description
sensor_id	int	Primary key for the sensor
sensor_description	varchar	A brief description of the sensor
sensor_name	varchar	Name of the sensor
installation_date	date	Date when the sensor was installed
sensor_status	varchar	Current status of the sensor
note	varchar	Additional notes
direction_1	varchar	Direction of the sensor
direction_2	varchar	Secondary direction of the sensor
latitude	float	Latitude of the sensor
longitude	float	Longitude of the sensor

location	varchar	Location of the sensor

Table: Counts

Column Name	Data Type	Description
ID	int	Primary key for the count
Date_Time	datetime	Date and time of the count
Year	int	Year of the count
Month	int	Month of the count
Date	int	Day of the count
Day	varchar	Day of the week of the count
Time	varchar	Time of the count
Hourly_Counts	int	Hourly pedestrian count
Sensor_ID	int	Foreign key to the Sensors table
Sensor_Name	varchar	Name of the sensor

Besides, this is a diagram created by dbdiagram for the above tables' schemas,



Conclusion

In this task, we have addressed several business requirements related to pedestrian counts, including showing the top 10 locations by day and by month, identifying the locations that have shown the most decline due to lockdowns in the last two years, and the locations that have shown the most growth in the last year. Additionally, we have loaded and staged the data in an appropriate format for future querying, and derived several metrics from the datasets, such as average hourly pedestrian counts by location and by direction, median hourly pedestrian counts by location, total pedestrian counts by day of the week and by month, and the busiest hour of the day by location. We have also suggested other metrics that could be derived from these data sets, such as the percentage change in pedestrian counts by location from one year to the next.

To accomplish these tasks, we have used two datasets containing information about sensors and pedestrian counts. We have merged the datasets on the sensor ID and Sensor_ID columns and converted the date_time column to a datetime object. We have then added several columns to the merged dataset, such as day of the week, year, month, date, and hour, to facilitate the analysis. We have used pandas to perform various groupings and aggregations on the data and visualized the results using plots.

Overall, this project has demonstrated how data analysis can help to gain insights into pedestrian counts, and how these insights can inform decision-making in areas such as urban planning and public safety. It has also highlighted the importance of data quality and data modelling in ensuring the accuracy and consistency of the results.