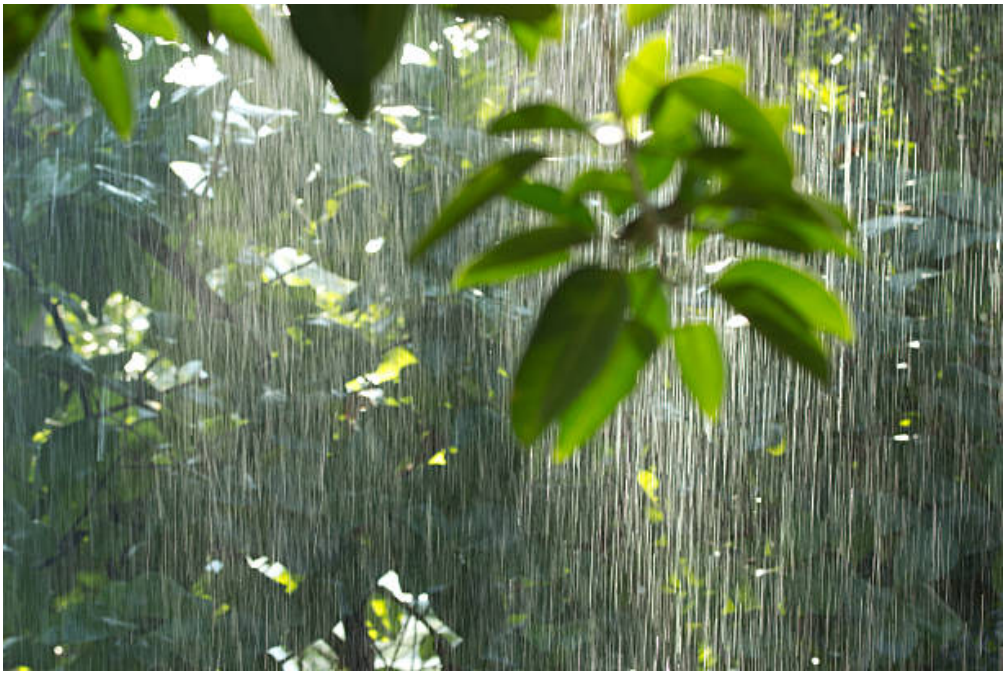

RAINFALL IN AUSTRALIA

Contents

1 Introduction

Australia is considered as a country with extreme weather conditions and different climatic patterns. This is known fact all over the world. Its weather is ranging from some of the driest areas on Earth to regions experiencing heavy rainfall periodically. This unique environment shows various challenges, limitations and opportunities for different sectors such as agriculture, urban planning and environmental conservation. Few examples are farmers needs to know about the rainfall patterns to decide when to plant crops and manage water usage, urban planners uses weather information to create buildings and structures that can handle both heavy rain and dry seasons. Steps that take to protect environment should consider rainy seasons. Knowing these rainy days helps to control and handle the water levels in rivers and lakes that supporting plant and animal life. It is necessary to ensure that every species have water for their survival. This is specially important in the regions where water levels can change quickly and affect the balance of local wildlife.

Understanding rainfall patterns and their associated climatic variables is very important for managing Australia's natural resources as there are both droughts and floods in some times. In 1997 there was a huge drought in in Australia called as The Millennium Drought. It was expanded over 500 000 square kilo meters in Souther Australia. It is said that this huge environmental hazard was ended by another environmental hazard that was a flood. This was also a widespread flood occured around 2010 and 2011. The millenium drought was a drought that was caused due to the lack of rainfall. But in some areas in Southern Australia got average rainfall time to time. In later 2011, climate conditions in some areas have returned to pre drought averages and there were other areas that remained below average. Due to these drougts there were huge water shortage and it impacts on farmers for their cultivations as they receive only limited water and there ware many times that water was restricted in cities. Water shortage led to limited water allocation for large investments in infrastructure and this paves the way of economic impact of 1.6% of Australia's gross domestic product [1].

Less rainfall has impacted on plants, animals and its ecosystem. In Southwest Western Australia fresh water fish species destroyed due to this harsh weather condition. Water security of Australia is threatened by the climatic changes. As the temperature goes up, the use of water get increased normally. Most people started to look for municiple water supplies as the domestic rain water tanks got dried up. This increased the demand on water and pressue on the authorities. After experiencing droughts for long time and huge rainfall happened with heavily. These sudden rains resulted the sediments and impurities in the atmosphere to add into the natural water resources like streams and lakes. This also a health warning as the immune system faces difficult in these kind of climatic changes. In Sydney around 2007, there was a bacteria that was causing health issues with water and its called as cyanobacteria. This was happened in Burragorang lake and caused by Cryptosporidium and Giardia [2]. Though the rainfall in Australia is so inconsistent, there is an great impact of it on water resources, agriculture, infrastructure and ecological systems. So it is necessary to study on rainfall trends and environmental factors. This is an important fact on various sectors in decision making and planning strategies.

The main objective of this project is to conduct an analysis of rainfall in Australia by considering different environmental factors that are influencing on rainfall in different locations and different time periods. Normally environmental conditions such as temperature, humidity and wind speed are interconnected with rainfall patterns. By focusing on these factors, this project aims to identify the patterns of rainfall that can be helpful for predicting and managing risks of having extreme weather conditions. This is using data warehousing and big data techniques to get the accurate output of the project.

In the second section, this report discuses about the motivation of the project like what kind of goals are planned as the output of the project. For this project, a csv file is used as the input dataset that consists of various environmental conditions. It contain weather metrics such as wind directions, rain fall amounts, temperature in different locations and different times. So the third section will give a more details on the dataset used for the project. In the next stage data extraction is needed to be done and it is mentioned the fourth section. A dataset should subjected to a quality check and cleaning is a necessary step. So this is presented in the fifth stage. As there were few issues and fixes done when the project was going on, those points are addressed in the sixth section. Project output will be analysis by exploring visualizations from powerBI and those things will be discussed in the seventh section. Overall project challenges and limitations are mentioned in the eighth section. Finally shows the Conclusion of the project.

2 Motivation

Due to the frequently changing weather conditions in Australia its very dificult to manage water, planning agriculture and preparing for extreme weather. Because of this inconsistent climate in Australia, it is important to understand how rainfall works in different parts of the country and how its connected to other factors such as temperature, humidity and wind. This project is motivated by this need of identifying weather patterns better, which helps not only the people in each and every categories, but also the animal species that need water and natural balance for their survival.

Major part of Australian economy is covered by the agriculture done in the country. When rainfall is low crops do not grow properly and on the other hand when there is a huge rainfall, then also the crops can be destroyed and soil can be washed away. By analyzing rainfall data connection with the temperature, wind and other conditions, this project has the ability of helping farmers to prepare for different climates. As an example, selecting crops that grow well even there are changes in the rainfall pattern.

In urban areas are highly affected by rainfall. As there are high population in the cities, it is necessary to have better planning before any of the weather condition. In relation to the water, there is a growing need of water supply and infrastructure that handle both flood and droughts. So if there is a way to identify these rain patterns, it is beneficial for the city planners and policy makers to take better decisions about the water storage and flood protection. If the authorities get accurate information on the rainfall, they can invest on the infrastructures that can implemented to prevent flooding and to ensure enough water supply during dry periods. Planning before flooding and droughts help to protect people and properties in the cities. To make better plans it necessary to identify rainfall patterns.

Preparing for extreme weather condition is another motivation for this project. There is an increase in extreme weather in Australia that ranges from droughts to heavy rain that causes floods. The output of this project can help to develop weather forecasts by examining conditions that frequently occur before heavy rain, such as specific wind speeds or humidity levels. This can give emergency teams more time to prepare and warn people in high risk areas.

On technical level, this project has an opportunity to use data warehousing and big data skills. Using database tool like MS SQL Server, programming language like Python and data visualization tool like Power BI, we can turn large number of data into clear and useful data. Data quality checks and data storage in a star schema give the chance to manage and analyze the data. These skills are highly valuable in these days and this project shows how they can be used to solve real world problems related to climate.

3 Proposed Solution

By considering various requirements it is identified that there is a need of knowing rainfall patterns in Australia. So I have proposed a solution based on data warehouse and big data concepts. Here I have used 4 major components to achieve the goals of the project. The diagram 1 below show system architecture for this project.

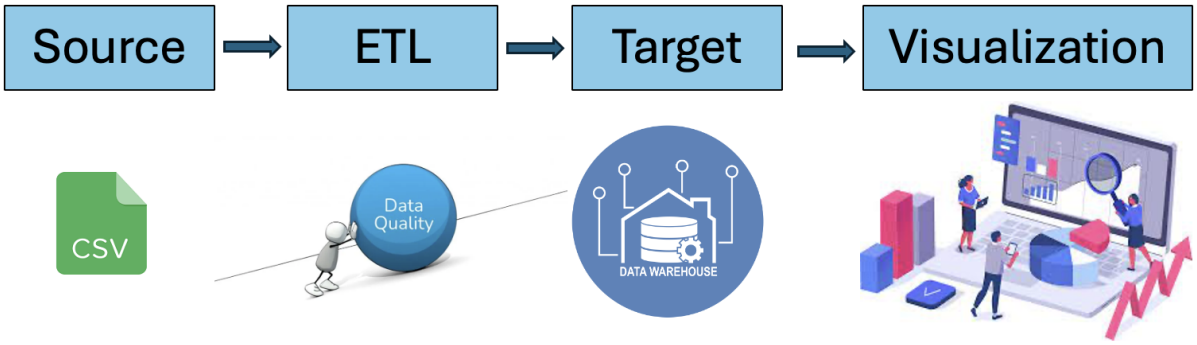

Figure 1: System Architecture

I have used several technologies in order to get the accurate out put of the project.
MacOS - As the operating system
Python - As the programming language
MS SQL server - To create the data warehouse
VScode - As the code editor
MS PowerBI - To visualize the data stored in the data warehouse
UTM Virtual machine - To create the windows virtual machine to install power BI
Docker - As the container that supports to run MS SQL server in IOS environment

4 Dataset

The input source is a csv file and this dataset was taken from kaggle website [3]and they have gathered data from the Bureau of Meteorology's online climate data, Canberra weather and etc. This dataset contains data of daily weather observations from Australian weather stations for about 10 years from 2007 to 2017. The target column is the RainTomorrow that says YES or NO. Simply it says that did it rain the next day. The dataset consist of 23 columns and 142 193 rows.

Name of the Variable	Data Type	Description
Date	Date	The date that observe the climatic conditions in the relevant area
Location	Plain text	The common name of place where the weather metrics were taken (Location of the station)
MinTemp	Number	The minimum temperature measured during that time in degrees celsius
MaxTemp	Number	The maximum temperature measured during that time in degrees celsius
Rainfall	Number	Rainfall amount measured during the specific day in mm
Evaporation	Number	The evaporation measured in 24 hours to 9am in mm
Sunshine	Number	The count of hours of sunshine in the day
WindGustDir	Plain text	The strongest wind direction gust in the 24 hours to midnight
WindGustSpeed	Number	The strongest wind gust speed (Kilometers per hour km/h) in the 24 hours to midnight
WindDir9am	Plain text	Wind Direction at the time of 9am
WindDir3pm	Plain text	Wind Direction at the time of 3pm
WindSpeed9am	Number	Average speed (km/hr) of wind before 10 minutes prior to 9am
WindSpeed3pm	Number	Average speed (km/hr) of wind over 10 minutes before to 3pm
Humidity9am	Number	Percentage of Humidity at the time of 9am
Humidity3pm	Number	Percentage of Humidity at the time of 3pm
Pressure9am	Number	Atmospheric pressure (hpa) reduced to mean sea level at the time of 9am
Pressure3pm	Number	Atmospheric pressure (hpa) reduced to mean sea level at the time of 3pm
Cloud9am	Number	Fraction of sky covered by cloud at the time of 9am. This is measured in "oktas"(unit of eigths)
Cloud3pm	Number	Fraction of sky covered by cloud (in "oktas"-unit of eigths) at 3pm. See Cloud9am for a description of the values
Temp9am	Number	Temperature (degrees Celsius) at 9am
Temp3pm	Number	Temperature (degrees Celsius) at 3pm
RainToday	Number	Boolean: 1 if precipitation (mm) in the 24 hours to 9am exceeds 1mm, otherwise 0
RainTomorrow	Number	The next day rain amount in mm. Used to create response variable RainTomorrow. A kind of measure of the "risk"

Table 1: Feature Description of the dataset

5 OLAP System

In this project, OLAP (Online Analytical Process) system can be considered as target data warehouse. Here there are no online transactional data processing, so the best storage solution is a data warehouse as it gives the ability to fast, flexible, and multi-dimensional analysis of large datasets, which is important for effective business intelligence and decision-making. There are several valuable benefits from OLAP systems. One of them is multi dimensional analysis that allows to store multiple dimensions like time, geography, customer, product etc. This means users can analyze data from different ways and can easily switch between views, which is difficult in traditional relational databases. Another important fact on OLAP system is that these are optimized for complex analytical queries. They have done a pre aggregation of data and stored it in a format that allows for rapid access to summary information. This leads to much faster response times compared to traditional databases. Another advantage is OLAP systems allows users to perform operations like drill-down (going from summary data to detailed data), roll-up (going from detailed data to summary), slice (viewing a single layer of data), and dice (viewing a subset in multiple dimensions). These capabilities make it easy to explore data and generate reports.

There are several OLAP schema available such as star schema, snowflake schema and fact constellation schema. Here the star schema is the simplest and widely used OLAP schema[4]. In star schema there is central fact table that can contain quantitative data like sales, revenue or quantity and there are dimension tables around the fact table which contain data like date, product, location or customer. Each dimension table is connected directly to the fact table using a foreign key. This schema is called as star because the final picture of the schema is shown as star with the fact table at the center and the dimension tables as the points. One of the main advantage of star schema is query performance since dimension tables are denormalized as they are not split into smaller tables and queries need few number of joins and can execute quickly. Specially this schema is good for data warehouse with heavy OLAP operations. The simplicity of this structure is highly smart as this schema helps people who are not familiar with complex database structures like business users and analysts to access and navigate the data with ease. Furthermore, the star schema performs well in the aggregation and summarization as each dimension is directly connected to the fact table. This enables a rapid, multi dimensional analysis. The simplicity of the schema provides an easy way to optimize and maintain by making it well suitable for reporting and business purposes in organizations. So I have decided to use star schema type to implement the data warehouse in this project.

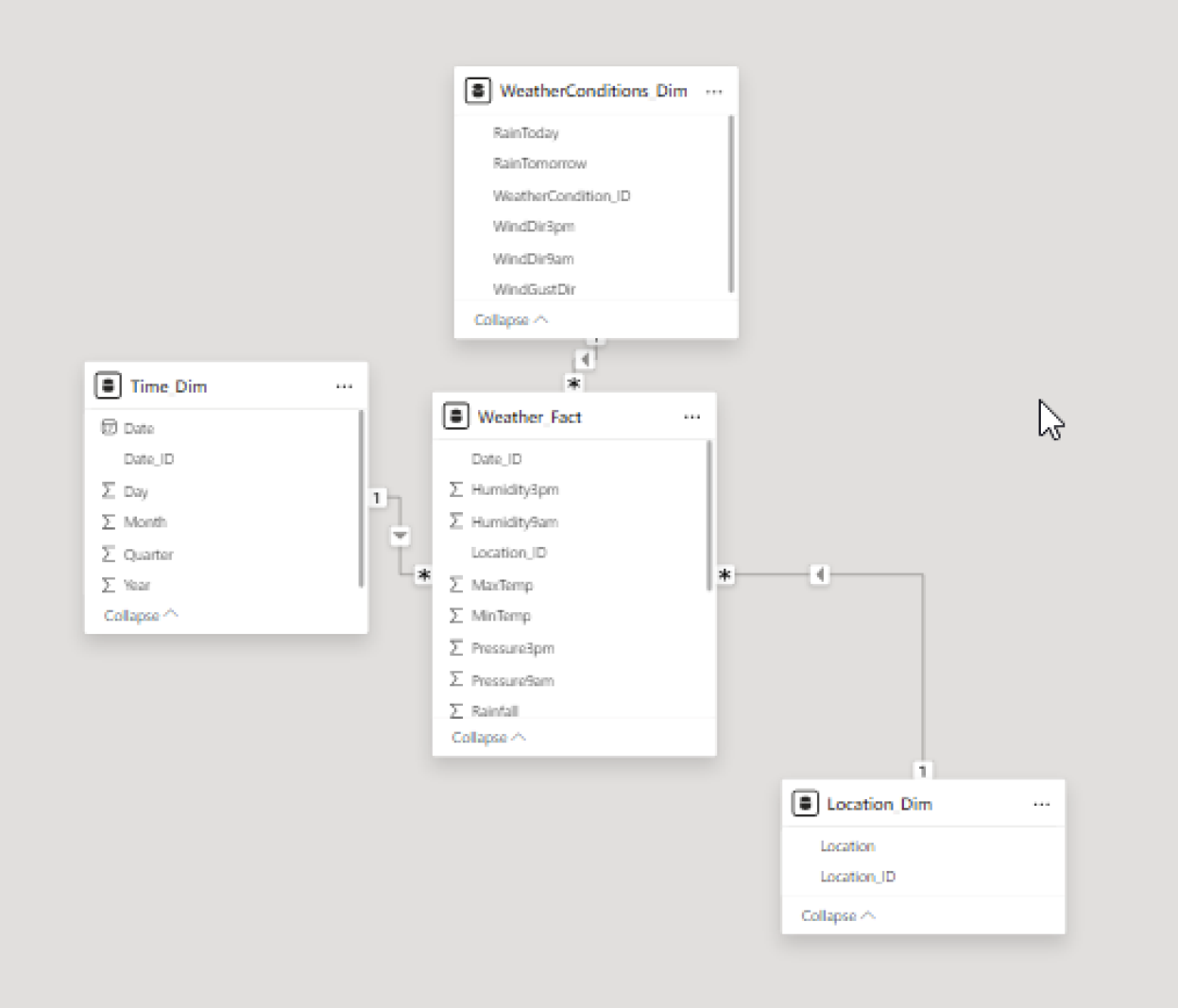

Figure 2: Star Schema

In this star schema there is one fact table as Weather_Fact and 3 dimension tables as Location_Dim, Time_Dim and WeatherConditions_Dim. This is implemented in the MS SQL server. Location is stored in the Location_Dim table and the primary key in this table is Location_ID. Time_Dim table contains attributes namely Date, Year, Month, Day and

Quarter. The primary key in Time_Dim table is Date_ID. WeatherConditions_Dim table consists of attributes namely RainToday, RainTomorrow, WindGustDir, WindDir9am and WindDir3pm. The primary key in WeatherConditions_Dim table is WeatherCondition_ID. In this star schema, fact table in the center is named as Weather_Fact and it consists of variables such as Location_ID, Date_ID, WeatherCondition_ID, MinTemp, MaxTemp, Rainfall, WindGustSpeed, WindSpeed9am, WindSpeed3pm, Humidity9am, Humidity3pm, Pressure9am, Pressure3pm, Temp9am and Temp3pm. The Location_ID in the fact table is the foreign key referencing the Location_ID in the Location_Dim table that represents as the primary key of Location_Dim table. The Date_ID in the fact table is the foreign key referencing the Date_ID in the Time_Dim table that represents as the primary key of Time_Dim table. The WeatherCondition_ID in the fact table is the foreign key referencing the WeatherCondition_ID in the WeatherConditions_Dim table that represents as the primary key of WeatherConditions_Dim table. Other attributes in the fact table stores the quantitative weather data that can be analyzed with various dimensions such as location, date, and weather conditions. This structure give the ability in efficient querying for trends and patterns in weather.

6 ETL

Now we have the source as dataset file and target as OLAP. So this dataset needs store into the database. But it needs go through the ETL process in order to prepare data as a accessible format.

6.1 Extraction

ETL refers to Extract, transform and Load and this is process that commonly involved in the data integration for the purpose of preparing data to analyze and report. As the first step here data extracted from the input source which can be spreadsheet, database or even real time data stream. In this project the data is extracted by the csv file using python programming language. Once it is extracted, the data goes through the transformation stage where the cleaning of data happen then data formatting and sometimes aggregated to ensure consistency and compatibility. There are several important steps are done in this stage. Few of them are cleaning of data and applying specific business rules to make the information more useful and meaningful.

6.2 Transform

6.2.1 Data Quality Checks and data cleaning

Data cleaning is an important step when preparing information for analysis. The process of data cleaning involves addressing several key area of data quality, such as completeness, accuracy, validity, uniformity and consistency[5]. By focusing on these areas, data cleaning ensures that datasets are not only usable but also able to produce meaningful and actionable insights.Complete data refers to that there are no missing values in the dataset provided as the input source. If there are missing values or gaps or empty cells in the spread sheets, those data can be lead to incorrect decisions. Missing values can be handled in different ways according the how many missing values are available with respect to the total number of values. For examples, missing values can be cleaned by filling with mean or median values for numeric data, or with mode or most frequent values for categorical data. In some cases as there are high percentage of missing values, rows or columns will be removed totally. Another most crucial requirement in data cleaning is data accuracy. Accurate data refers that the values in the dataset are correct and represent real world scenarios. There can be different ways of having incorrect data such as by human errors, faulty sensors, mistakes done when entering data and other issues happening when gathering data. During the process of data cleaning, these incorrect data needs to be identified and corrected. These steps can involved in comparing data against known sources, applying rules to identify outliers or using algorithms to detect and correct likely errors. Data validity is another key aspect of data cleaning that conforms to a defined format or set of rules. For example, there can be date values with different date formats such YYYY-MM-DD or DD-MM-YYYY. So these format should be changed to YYYY-MM-DD to get validated data in the date column. Also there can be invalid categorical values and these need to be changed into some predefined values. Invalid data can be there due to data entry errors, technical mistakes or by incompatible systems. During the process of cleaning, data validation rules are applied to ensure that all entries meet the required formats and constraints. Any values that are not follow these rules need to be corrected or removed. Data Uniformity means that data is given in a standardized format. This includes using consistent units of measurement, date formats and naming conventions in the dataset. As an example, temperature can be given in either Celsius or Fahrenheit in the dataset, but all values should be in only one form. Another instance is having different forms of naming conventions and these data needs to be cleaned by using a consistent naming style for columns or categorical values. During the process of data cleaning, converting non uniform data into uniform data standardizes to eliminate any conflicts. Other aspect of data cleaning is consistency and this refers to the alignment of values for related fields and records. Consistent data makes sure that there are no

conflicts in the dataset. For example, if one field showed that a product was sent on a specific date, other fields should not show conflicting information, such as delivery occurring before shipment.

In this project also the input dataset is subjected to some quality checks and data cleaning process.

- **Missing Values - Check and Cleaning**

First check is done to identify that number of missing values available in each columns 3. Here total number of missing values and percentage value is counted 4. Then its giving an idea on columns with empty columns and what steps need to take for cleaning process.

```python
def check_blank_cells(data):

    print("\nColumns with Blank Cells:")
    blank_cells_count = data.isna().sum()
    total_rows = len(data)

    blank_columns = blank_cells_count[blank_cells_count > 0]
    blank_percentages = (blank_columns / total_rows) * 100

    blank_summary = pd.DataFrame({
        'Blank Cells Count': blank_columns,
        'Percentage of Blanks (%)': blank_percentages
    })

    print(blank_summary)
```

Figure 3: Data Quality Checks for missing values

```
Columns with Blank Cells:
               Blank Cells Count  Percentage of Blanks (%)
MinTemp                     1485                  1.020899
MaxTemp                     1261                  0.866905
Rainfall                    3261                  2.241853
Evaporation                62790                 43.166506
Sunshine                   69835                 48.009762
WindGustDir                10326                  7.098859
WindGustSpeed              10263                  7.055548
WindDir9am                 10566                  7.263853
WindDir3pm                  4228                  2.906641
WindSpeed9am                1767                  1.214767
WindSpeed3pm                3062                  2.105046
Humidity9am                 2654                  1.824557
Humidity3pm                 4507                  3.098446
Pressure9am                15065                 10.356799
Pressure3pm                15028                 10.331363
Cloud9am                   55888                 38.421559
Cloud3pm                   59358                 40.807095
Temp9am                     1767                  1.214767
Temp3pm                     3609                  2.481094
RainToday                   3261                  2.241853
RainTomorrow                3267                  2.245978
```

Figure 4: Output of missing values check

As a result of checking it can be seen that are few columns such as Evaporation, Sunshine, Cloud9am, Cloud3pm have many missing values. As the percentages are more than 35% of the total number of rows. So its highly impacting on the final investigations as its around one third of the whole column rows. So I have decided to drop off these columns as the cleaning step and in further steps of data cleaning will explain the cleaning process of other missing values 5.

```
def check_null_percent_and_drop_blank_columns(data, threshold=35):
    blank_cells_count = data.isna().sum()
    total_rows = len(data)
    blank_columns = blank_cells_count[blank_cells_count > 0]
    blank_percentages = (blank_columns / total_rows) * 100
    columns_to_drop = blank_percentages[blank_percentages > threshold].index.tolist()
    data = data.drop(columns=columns_to_drop)
    print(f"Dropped columns with more than {threshold}% blanks: {columns_to_drop}")
    return data
```

Figure 5: Dropping columns with more than 35% of missing values

- **Invalid Direction - Check and Cleaning**

In this data set there are few columns that gives the directions. These cells have the values of wind direction for different times and they are in plain text. The name of these columns are WindGustDir, WindDir9am, WindDir3pm. Here the directions are named as "N" for North, "E" for East, "W" for West and "S" for South. So values can have multiple letters but these columns should contain these 4 letters only. So I did a check to find out whether there are other letters or not. And also I also retrieved the missing values in this check 6.

```python
def count_invalidDirection_and_blank_rows(data):
    print("\nCheck for letters other than N,E,W,S in WindGustDir, WindDir9am and WindDir3pm columns:")

    # Define the allowed characters
    allowed_chars = {'N', 'E', 'W', 'S'}

    # Initialize counters for rows with invalid entries and blank cells
    invalid_row_count = 0
    blank_row_count = 0

    # Columns to check for invalid letters and blank cells
    columns_to_check = ['WindGustDir', 'WindDir9am', 'WindDir3pm']

    # Function to check if a cell value contains only allowed characters
    def is_valid_direction(value):
        # Return False if any character in the cell is outside the allowed set
        return set(str(value)).issubset(allowed_chars)

    # Check each row for invalid letters and blank cells in the specified columns
    for index, row in data.iterrows():
        # Flags for tracking invalid and blank cells in each row
        has_invalid = False
        has_blank = False

        # Check each specified column in the row
        for col in columns_to_check:
            cell_value = str(row[col]).strip()
            if cell_value == '' or pd.isna(row[col]):  # Check for blank cells
                has_blank = True
            elif not is_valid_direction(cell_value):  # Check for invalid letters
                has_invalid = True

        # Update counters based on flags
        if has_blank:
            blank_row_count += 1
        if has_invalid:
            invalid_row_count += 1

    # Print results
    print(f"Rows with blank cells in specified columns: {blank_row_count}")
    print(f"Rows with invalid letters in specified columns: {invalid_row_count}")
```

Figure 6: Data Quality Checks for invalid directions

```
Check for letters other than N,E,W,S in WindGustDir, WindDir9am and WindDir3pm columns:
Rows with blank cells in specified columns: 18620
Rows with invalid letters in specified columns: 2
```

Figure 7: Output of invalid directions check

In this check it is identified that there are 2 rows with invalid letters and 18620 rows with missing values. So these things need to be cleaned up to make sure that there are no conflicts in the dataset when storing in the data warehouse. For clean up step I have decided to remove those two rows from the dataset. Furthermore, I have decided to replace the missing values with NULL.

```python
def count_and_clean_direction_data(data):
    columns_to_check = ['WindGustDir', 'WindDir9am', 'WindDir3pm']
    allowed_chars = {'N', 'E', 'W', 'S'}
    rows_to_drop = []

    def is_valid_direction(value):
        return set(str(value)).issubset(allowed_chars)

    for index, row in data.iterrows():
        if any(not is_valid_direction(row[col]) for col in columns_to_check):
            rows_to_drop.append(index)

    data = data.drop(index=rows_to_drop)
    data[columns_to_check] = data[columns_to_check].applymap(lambda x: "NULL" if pd.isna(x) or str(x).strip() == '' else x)
    print(f"Dropped rows with invalid direction data: {len(rows_to_drop)} rows dropped.")
    return data
```

Figure 8: Removing rows & replacing with NULL

- **Humidity Values - Check and Cleaning**

As the next step of quality check, I have focused on the columns that containing humidity values. Normally humidity values should be in between 0 and 100. Though there are two columns namely "Humidity3pm" and "Humidity9am", the impact of these values on the final data reports is high. If there are values outside of this range can be considered as invalid data. And also I have done a check for missing values in these 2 columns with this check as these too need to be cleaned.

```python
# Humidity Range Check
def humidity_count_invalid_and_blank_rows(data):
    print("\nRange Checks for Humidity columns:")

    invalid_number_count = 0
    blank_row_count = 0

    columns_to_check = ['Humidity9am', 'Humidity3pm']

    for index, row in data.iterrows():

        has_invalid = False
        has_blank = False

        for col in columns_to_check:
            cell_value = row[col]

            if pd.isna(cell_value):
                has_blank = True
            elif not (0 <= cell_value <= 100):
                has_invalid = True

        if has_blank:
            blank_row_count += 1
        if has_invalid:
            invalid_number_count += 1

    # Print results
    print(f"Rows with blank cells in specified columns: {blank_row_count}")
    print(f"Rows with invalid numbers in specified columns: {invalid_number_count}")
```

Figure 9: Data Quality Checks for humidity values

```
Range Checks for Humidity columns:
Rows with blank cells in specified columns: 5274
Rows with invalid numbers in specified columns: 1
```

Figure 10: Output of humidity values check

As a result of this quality check it is identified that there is one row with invalid value and 5274 rows with missing values. In the clean up process I have decided to check the nearest outlier of the invalid number and bring to that outlier. As an example, if there numbers more than 100 and those will brought to their nearest outlier 100. If there numbers less than 1 and those will brought to their nearest outlier 1. Furthermore in the cleaning, missing values are replaced with 0 as these columns contain numerical values.

```
def humidity_round_off_outliers_and_replace_blanks(data):

    columns_to_check = ['Humidity9am', 'Humidity3pm']
    def process_value(value):
        if pd.isna(value) or str(value).strip() == '': return 0
        elif value < 0: return 0
        elif value > 100: return 100
        else: return value
    for col in columns_to_check:
        data[col] = data[col].apply(process_value)
    print("Rounded outliers and replaced blanks with 0 in Humidity columns.")
    return data
```

Figure 11: Rounding off to outliers & replacing with 0

- **Date Values - Check and Cleaning**

The next check is for the date column and quality checking on the date column in a dataset is a crucial step to ensure data accuracy and consistency. This step includes the checking on data format such that its necessary to have all the date values in a one standardized format like YYYY-MM-DD to get a accurate analysis without conflicts. Other point is checking for future dates to know whether this date is not yet come. Furthermore, I have checked for the missing values in this columns, as if there are row with missing dates it will harm the output so badly.

```python
# 4- Date format and future dates

def check_date_column(data, date_column='Date'):

    data = pd.read_csv(file_path)

    blank_count = 0
    invalid_format_count = 0
    future_date_count = 0

    for value in data[date_column]:
        if pd.isna(value) or str(value).strip() == '':
            blank_count += 1
        else:
            try:
                parsed_date = datetime.strptime(value, "%Y-%m-%d")

                if parsed_date > datetime.now():
                    future_date_count += 1
            except ValueError:

                invalid_format_count += 1

    # Print results
    print(f"Blank cells in '{date_column}': {blank_count}")
    print(f"Invalid date format count in '{date_column}': {invalid_format_count}")
    print(f"Future date count in '{date_column}': {future_date_count}")
```

Figure 12: Data Quality Checks for date column

```
Blank cells in 'Date': 0
Invalid date format count in 'Date': 1
Future date count in 'Date': 0
```

Figure 13: Output of date values check

As a result of this quality check on date column, I was able to ensure that there are no missing values in this dataset. With related to the invalid date format checking there is one cell that contained invalid date format that requires cleaning. Then regarding future dates checking, it was not able to identify any of the future date values. For the cleaning step here, I had to pay attention only to date format cleaning and I decided to change the existing invalid format into YYYY-MM-DD standard formation.

```python
def convert_date_format(data, date_column='Date'):
    def parse_date(value):
        try:
            return datetime.strptime(value, "%Y-%m-%d").strftime("%Y-%m-%d")
        except ValueError:
            for fmt in ("%d/%m/%Y", "%d-%m-%Y", "%m/%d/%Y", "%Y/%m/%d"):
                try:
                    return datetime.strptime(value, fmt).strftime("%Y-%m-%d")
                except ValueError:
                    continue
            return "NULL"
    data[date_column] = data[date_column].astype(str).apply(parse_date)
    print("Converted Date column to 'YYYY-MM-DD' format.")
    return data
```

Figure 14: Date format changing

- **RainToday and RainTomorrow Values - Check and Cleaning**

The next columns I was focusing were RainToday and RainTomorrow columns as there are values only as "YES" and "NO". So I did a quality check to identify the whether there areinvalid values other than YES and NO in these columns. And also count of missing values is taken with this quality check as it is an important step to minimize issues and conflicts.

```python
def check_rain_columns(data):

    print("\nChecking 'RainToday' and 'RainTomorrow' columns for valid values:")

    data = pd.read_csv(file_path)

    columns_to_check = ['RainToday', 'RainTomorrow']

    allowed_values = {'YES', 'NO'}
    invalid_counts = {col: 0 for col in columns_to_check}
    blank_counts = {col: 0 for col in columns_to_check}


    for col in columns_to_check:
        for value in data[col]:
            if pd.isna(value):
                blank_counts[col] += 1
            elif str(value).strip().upper() not in allowed_values:
                invalid_counts[col] += 1

    for col in columns_to_check:
        print(f"\nColumn: {col}")
        print(f"Blank cells: {blank_counts[col]}")
        print(f"Invalid cells: {invalid_counts[col]}")
```

Figure 15: Data Quality Checks for RainToday and RainTomorrow columns

```
Checking 'RainToday' and 'RainTomorrow' columns for valid values:

Column: RainToday
Blank cells: 3261
Invalid cells: 0

Column: RainTomorrow
Blank cells: 3267
Invalid cells: 3
shelomi@Shelomis-MacBook-Pro climate %
```

Figure 16: Output of RainToday and RainTomorrow values check

As result of this quality check I was able to see that there are 3261 missing values and no other invalid values in RainToday column and there are 3267 missing values and 3 invalid values for these columns in the dataset. During the cleaning process of this, invalid data and missing values both are replaced by NULL.

```python
def check_and_replace_rain_columns(data):
    columns_to_check = ['RainToday', 'RainTomorrow']
    allowed_values = {'YES', 'NO'}
    for col in columns_to_check:
        data[col] = data[col].apply(lambda x: "NULL" if pd.isna(x) or str(x).strip().upper() not in allowed_values else
    print("Replaced invalid values in RainToday and RainTomorrow columns with 'NULL'.")
    return data
```

Figure 17: Invalid data changing & replacing missing values

Other than these columns that were subjected to cleaning, there were 11 other columns left such as Location, MinTemp, MaxTemp, Rainfall, WindGustSpeed, WindSpeed9am, WindSpeed3pm, Pressure9am, Pressure3pm, Temp9am, Temp3pm. I have noted that all these columns consist of numeric values. As the final step of the cleaning process I was able to check for the missing values in these columns and I have decided replace all the missing values with 0 value.

7 Load

Then this clean csv file sepated to 4 csv files as Location_Dim.csv, Time_Dim.csv, WeatherCondition_Dim.csv and Fact_Weather.csv according to the tables created in the database using python scripts. The next step is to loading each csv file for the specific table. Location_Dim.csv loaded into Location_Dim, Time_Dim.csv file loaded into Time_Dim table, WeatherCondition_Dim.csv file loaded into WeatherCondition_Dim table 18 and Fact_Weather.csv file loaded into Fact_Weather table. Loading csv files into the MS SQL server is also done by using python. Below image of loading WeatherCondition_Dim.csv file loaded into WeatherCondition_Dim table is an example python code.

```python
Dim_WeatherConditions.py > ...
import pyodbc
import pandas as pd


conn = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL Server};SERVER=localhost;DATABASE=WeatherDB;UID=SA;PWD=[redacted]')
cursor = conn.cursor()


file_path = '/Users/shelomi/Documents/UNITEC/Data Warehouse and Big data/Assignment2/dataset/files/WeatherCondition_Dim.csv'
weather_condition_data = pd.read_csv(file_path, usecols=['RainToday', 'RainTomorrow', 'WindGustDir', 'WindDir9am', 'WindDir3pm'])

for index, row in weather_condition_data.iterrows():
    cursor.execute("""
    INSERT INTO WeatherConditions_Dim (RainToday, RainTomorrow, WindGustDir, WindDir9am, WindDir3pm)
    VALUES (?, ?, ?, ?, ?)
    """, row.RainToday, row.RainTomorrow, row.WindGustDir, row.WindDir9am, row.WindDir3pm)
conn.commit()

# Close the connection
cursor.close()
conn.close()

print("Data loaded successfully into WeatherCondition_Dim table.")

```

Figure 18: Load data to WeatherDim table

8 Exploring and Analysing Data

After the data loading into the database, the next step is to integrate MS PowerBI with MS SQL server. Since I am using mac opearting system for the project, I had to create a windows virtual machine to install MS PowerBI. After this connection, many charts and reports can be generated in the PowerBI dashboard to analyze data.

This pie chart is showing the locations that get highest amount of rainfall namely Caims, Darwin, CoffsHarbour, GoldCoast and Wollongong. The pie chart is showing the sum of rainfall and as a percentage value.

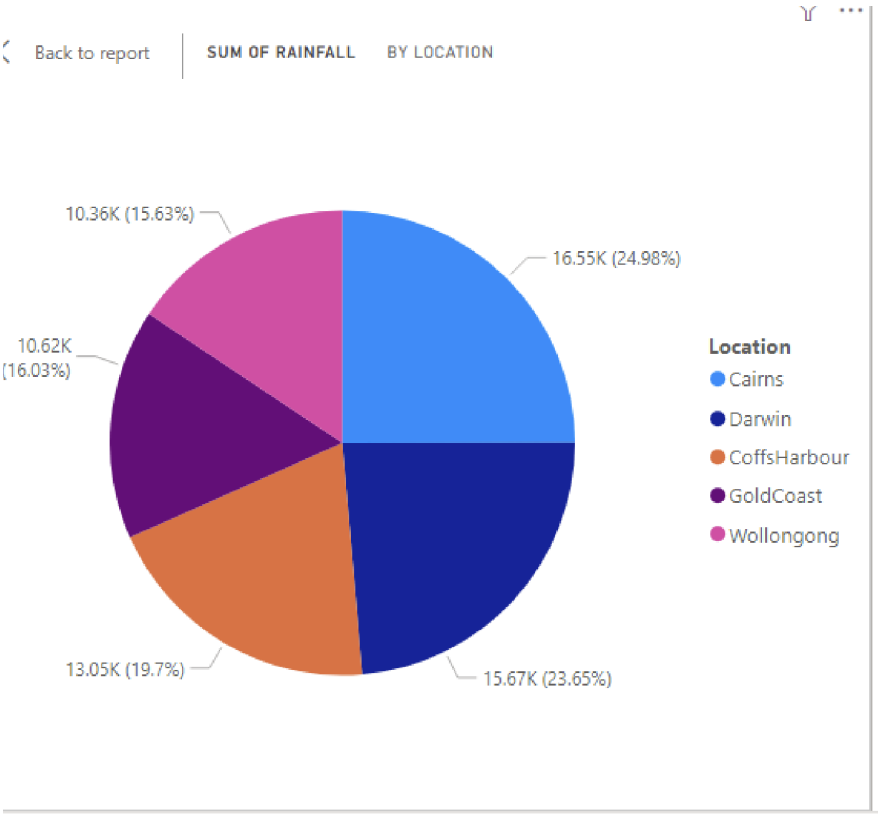

Figure 19: Locations with highest amount of rainfall

This below bar graph shows the sum of rainfall by year. It can be seen that in year 2010, 2011 and 2016 are the years that have received highest amount of rainfall.

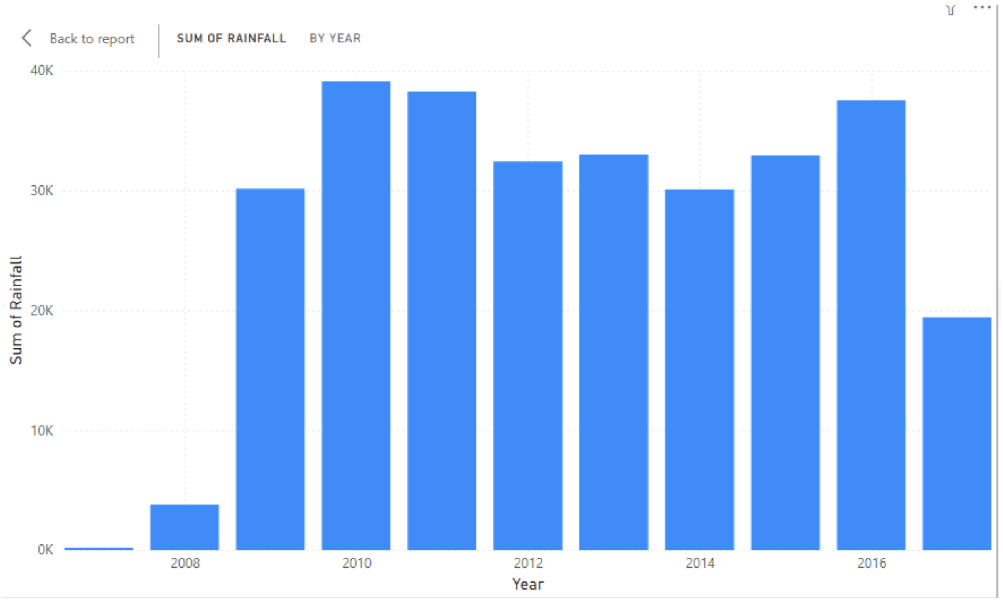

Figure 20: Rainfall by Year

This below line chart is showing the sum of temperature at 3pm in the top line, sum of temperature at 9am in the middle line and sum of rainfall in the bottom line. It can be seen that there is a slight increase in rainfall with the increase of temperature and slight decrease in rainfall with the decrease of temperature. So there is a slight impact of temperature on rain.

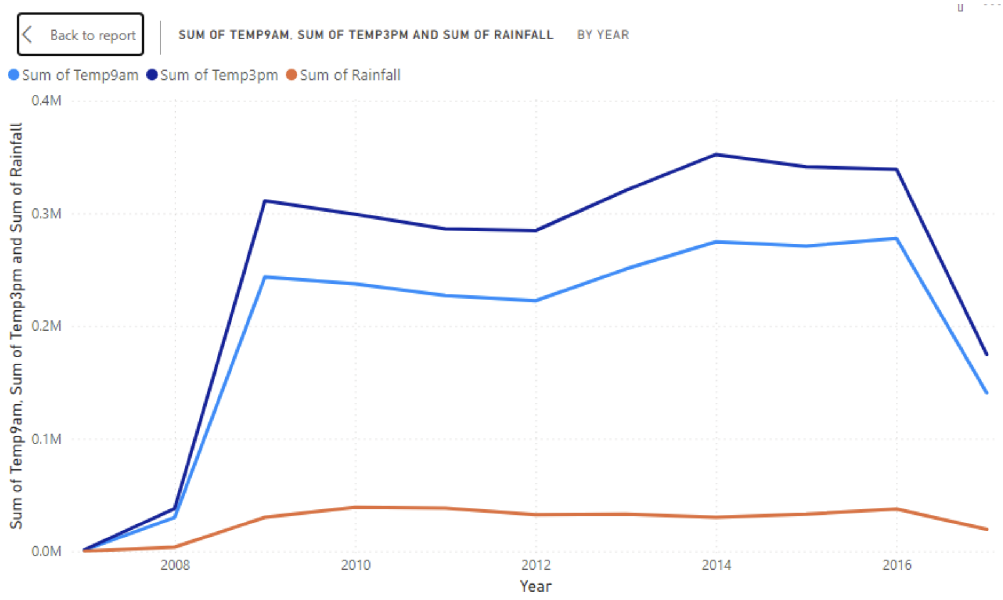

Figure 21: Temperature at 9am, Temperature at 3pm, Rainfall by Year

This below line chart is showing the sum of humidity at 9am in the top line, sum of humidity at 3pm in the middle line and sum of rainfall in the bottom line. It can be seen that there is a very less increase in rainfall with the increase of humidity and very less decrease in rainfall with the decrease of humidity. So there is a lease impact of humidity on rain.

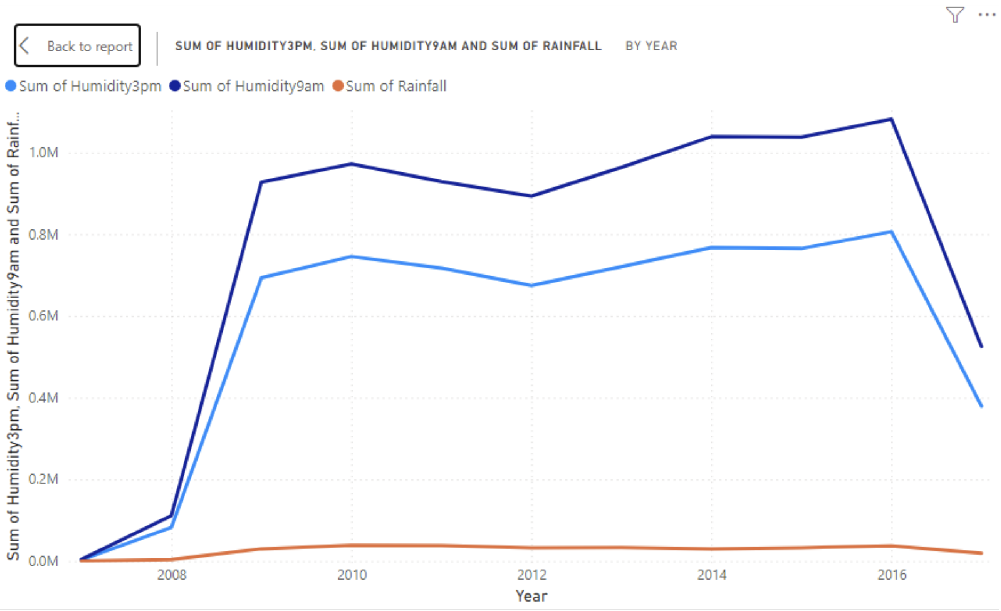

Figure 22: Humidity at 3pm, Humidity at 9am, Rainfall by Year

This below bar graph shows the sum of rainfall and sum of max temperature by rain today. It can be seen that No Rain today has very high value with very low amount of temperature. So raining today has very less probability when there is very high temperature.

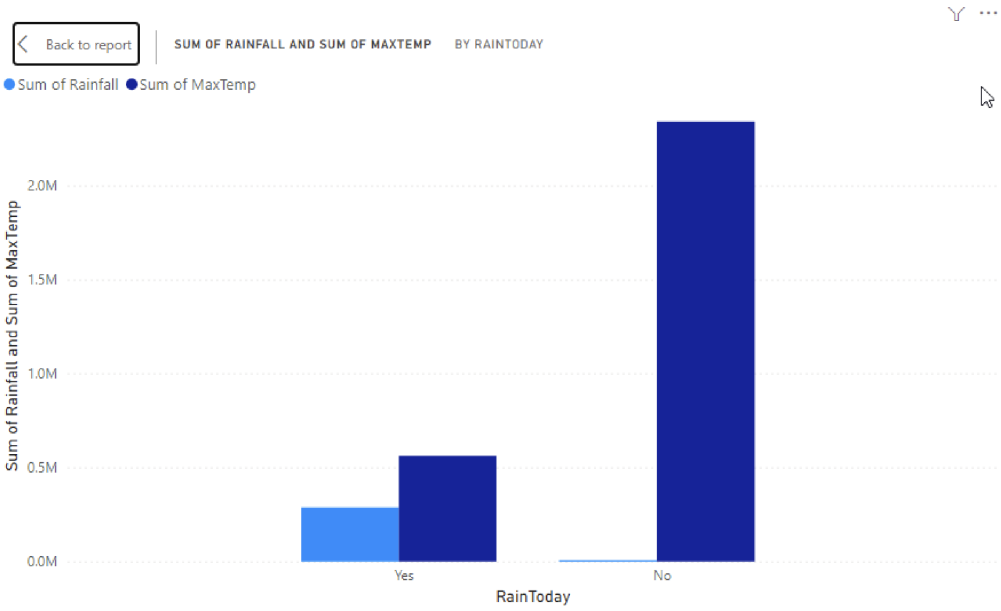

Figure 23: Rainfall and max temperature by RainToday

This below line chart is showing the sum of WindSpeed at 3pm in the top line, sum of WindSpeed at 9am in the middle line and sum of rainfall in the bottom line. It can be seen that there is an increase in rainfall with the decrease of WindSpeed and decrease in rainfall with the increase of WindSpeed. So there is an opposite impact of WindSpeed on rain.

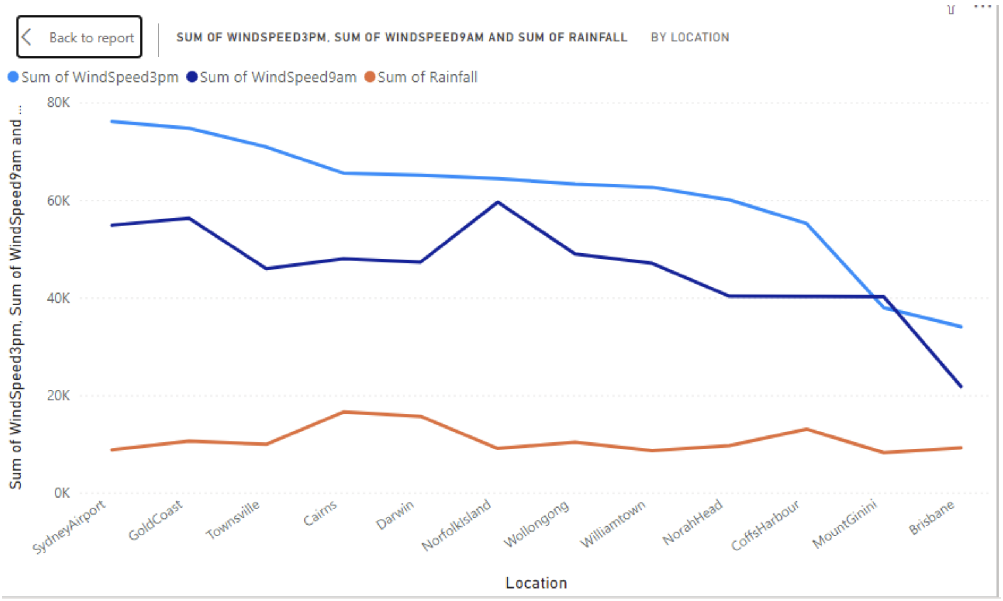

Figure 24: Wind Speed at 3pm, Wind Speed at 9am, Rainfall by Locations

9 Challenges And Limitations

9.1 Challenges

There were several challenges when loading data into the data warehouse. Most challenging one was with the Dim_WeatherConditions table. I have faced issues for several times in loading WeatherCondition_Dim.csv into WeatherConditions table.

```
shelomi@Shelomis-MacBook-Pro climate % python3 Dim_WeatherConditions.py
Error loading data from /Users/shelomi/Documents/UNITEC/Data Warehouse and Big data/Assignment2/dataset/Dim_WeatherConditions_after_cleanup.csv into Dim_WeatherConditi
ons: ('42000', '[42000] [Microsoft][ODBC Driver 17 for SQL Server][SQL Server]The incoming tabular data stream (TDS) remote procedure call (RPC) protocol stream is inc
orrect. Parameter 5 (""): The supplied value is not a valid instance of data type float. Check the source data for invalid values. An example of an invalid value is da
ta of numeric type with scale greater than precision. (8023) (SQLExecDirectW)')
shelomi@Shelomis-MacBook-Pro climate % python3 cleaning.py
Cleaning complete. Check the cleaned file at: /Users/shelomi/Documents/UNITEC/Data Warehouse and Big data/Assignment2/dataset/Dim_WeatherConditions_after_cleanup.csv
shelomi@Shelomis-MacBook-Pro climate % python3 Dim_WeatherConditions.py
Error loading data from /Users/shelomi/Documents/UNITEC/Data Warehouse and Big data/Assignment2/dataset/Dim_WeatherConditions_after_cleanup.csv into Dim_WeatherConditi
ons1: ('42000', '[42000] [Microsoft][ODBC Driver 17 for SQL Server][SQL Server]The incoming tabular data stream (TDS) remote procedure call (RPC) protocol stream is in
correct. Parameter 5 (""): The supplied value is not a valid instance of data type float. Check the source data for invalid values. An example of an invalid value is d
ata of numeric type with scale greater than precision. (8023) (SQLExecDirectW)')
shelomi@Shelomis-MacBook-Pro climate % python3 cleaning.py
Cleaning complete. Check the cleaned file at: /Users/shelomi/Documents/UNITEC/Data Warehouse and Big data/Assignment2/dataset/Dim_WeatherConditions_after_cleanup.csv
shelomi@Shelomis-MacBook-Pro climate % python3 cleaning.py
Cleaning complete. Check the cleaned file at: /Users/shelomi/Documents/UNITEC/Data Warehouse and Big data/Assignment2/dataset/Dim_WeatherConditions_after_cleanup.csv
```

Figure 25: Error on special characters

When I check the csv file there were no missing values, spaces or special characters that can be seen by eyes. Then I tried to find the column that causing the issue by creating csv files for every single column and tried to load those csv files to test weather tables created with single column table. Then I was able to identify that the issue was with the second column WindGustDir in the WeatherCondition_Dim.csv file.

But there was no invalid special values that can be seen by seeing. Then I assumed that there can be invalid characters that can not be seen by eyes and wrote a python code to remove all special characters.

```python
#Removing special characters in Dim_WeatherConditions
def clean_csv_special_char(input_file_path, output_file_path):
    with open(input_file_path, 'r') as infile, open(output_file_path, 'w', newline='') as outfile:
        reader = csv.reader(infile)
        writer = csv.writer(outfile)

        for row in reader:
            # Clean each cell in the row
            cleaned_row = [re.sub(r'[^a-zA-Z0-9\s]', '', cell) for cell in row]
            writer.writerow(cleaned_row)
```

Figure 26: Removing special characters

Then the data got extracted to Dim_WeatherConditions table.

9.2 Limitations

One significant limitation is the temporal range in data, which only covers a specific period. So this is missing newer patterns or long term trends in climate change. Another limitation can be related to the data quality as the quality of the data itself can impact results. As an example, missing values and outliers were addressed during data cleaning, but subtle data inaccuracies may still exist.

10 Conclusion

In conclusion, this project is implemented a data warehousing solution using the ETL process and an OLAP system to analyze rainfall patterns in Australia. The project shows the importance of data warehousing and big data techniques for analyzing large datasets. Despite some limitations, such as data coverage and scalability, the system creates a solid base for future improvements. Future work can expand the amount of data used, improve data quality checks, and add real time data processing to make weather analysis and climate research in Australia more detailed and effective.

References

[1] K. Fowler, M. Peel, M. Saft, T. J. Peterson, A. Western, L. Band, C. Petheram, S. Dharmadi, K. S. Tan, L. Zhang, *et al.*, "Explaining changes in rainfall–runoff relationships during and after australia's millennium drought: a community perspective," *Hydrology and Earth System Sciences*, vol. 26, no. 23, pp. 6073–6120, 2022.

[2] W. Steffen, R. Vertessy, A. Dean, L. Hughes, H. Bambrick, J. Gergis, and M. Rice, *Deluge and drought: Australia's water security in a changing climate*. Climate Council of Australia, 2018.

[3] J. Young, "Rain in australia."

[4] J. Pokorny, "Conceptual modelling in olap," in *Proc. Of 6th European Conference on Information Systems (ECIS98),(Ed. WRJ Baets), Aixen-Provence*, pp. 273–288, 1998.

[5] R. R. Pansara, "Master data management important for maintaining data accuracy, completeness & consistency," *Authorea Preprints*, 2023.