TITLE

Project Report

SUPERVISOR: Dr. Vibha Gaur

SUBMITTED BY

Nitesh Kumar (19001570023)

Ravinder Kumar (19001570030)

2021

Department of Computer Science

ACHARYA NARENDRA DEV COLLEGE

ACKNOWLEDGEMENT

This Project was jointly undertaken by **Nitesh Kumar** and **Ravinder Kumar** as their Semester-IV **Software Engineering Project**, under the guidance and supervision of ***Dr.* Vibha Gaur**. Our primary thanks goes to her, who poured over every inch of our project with painstaking attention and helped us throughout the working of the project. It is our privilege to acknowledge our deepest gratitude to her for her inspiration which has helped us immensely. We are extremely grateful to her for her unstilted support and encouragement in the preparation of this project.

Nitesh Kumar **Ravinder Kumar**

ACHARYA NARENDRA DEV COLLEGE
(University of Delhi)

CERTIFICATE

This is to certify that the project entitled "**Road Accident Data Analysis** " has been done by: **Nitesh Kumar** and **Ravinder Kumar** of Bachelor of Computer Science (Hons.) during Semester-IV from Acharya Narendra Dev College , University of Delhi under the Supervision of **Dr. Vibha Gaur.**

Nitesh Kumar **Ravinder Kumar**

Dr. Vibha Gaur

Supervisor

Contents

Chapter 1

PROBLEM STATEMENT

The present absence of extensive data systems equipped for gathering, classifying and detailing the accident and non-crash related injury information seriously confines the capacity to create, test, and implement alleviation strategies. The errand of recognizing injury causative components gets awfully theoretical without timely, accurate, complete, integrated, and available information that incorporates area, cause, contributing elements, and related activities associated with injuries involving personnel.

Government officials and general public are lacking systems which can show:-

1. What is the road accident ratio according to conditions?
2. In which zone the no. of accidents be maximum?
3. What is the death ratio during 2014-2017?
4. What is the no. of injured peoples zone wise?
5. What is the severity of these accidents?

Chapter 2

PROCESS MODEL

Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle. In this model, each module goes through the requirements, design, implementation and testing phases. Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieved.

A Incremental model is used for this account.

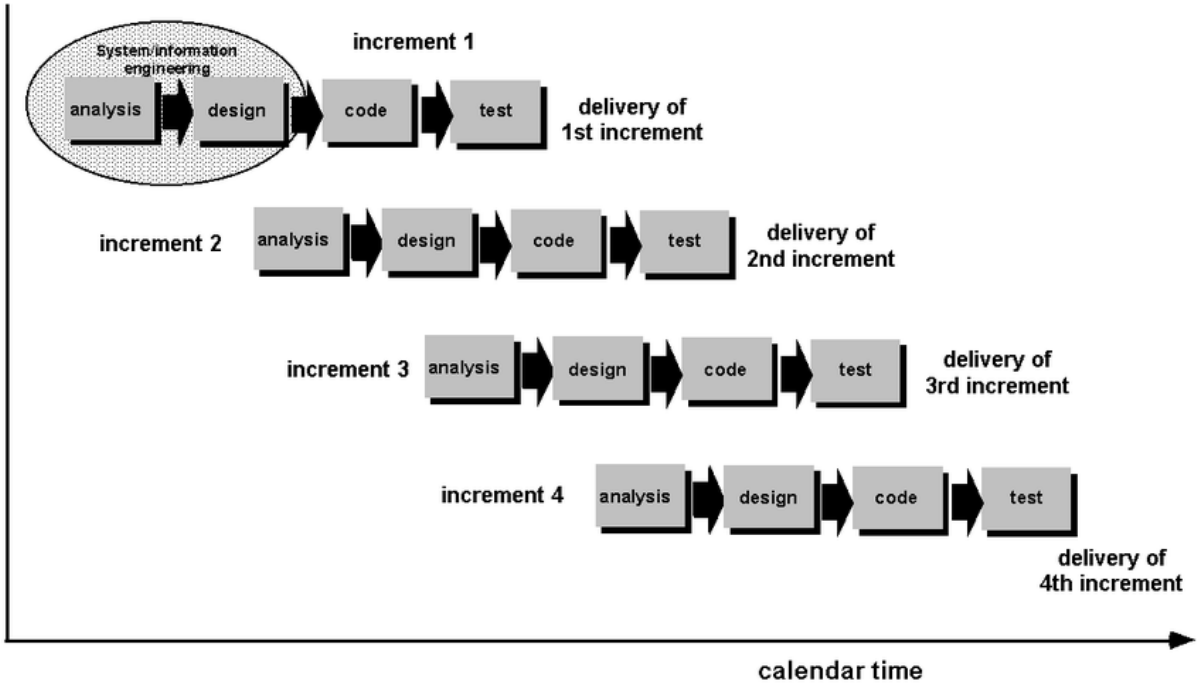


Figure 2.1 – Process Model
Image source:- https://www.researchgate.net

Advantages of using Incremental model for our project :-

- Errors are easy to be recognized.
- Easier to test and debug
- More flexible.
- Simple to manage risk because it handled during its iteration.
- The Client gets important functionality early.

Disadvantages of Incremental model :-

❖ Need for good planning

- ❖ Total Cost is high.
- ❖ Well defined module interfaces are needed.

Phases of Incremental model :-

1. Requirement analysis.
2. Design and Development.
3. Testing.
4. Implementation.

Chapter 3

REQUIREMNT ANALYSIS AND MODELING

3.1 **DFD**

DFD is the abbreviation for **Data Flow Diagram**. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart.

3.1.1 **Context level DFD**

A context level DFD is the most basic form of DFD. It aims to show how the entire system works at a glance. There is only one process in the system and all the data flows either into or out of this process. Context level DFD's demonstrates the interactions between the process and external entities.

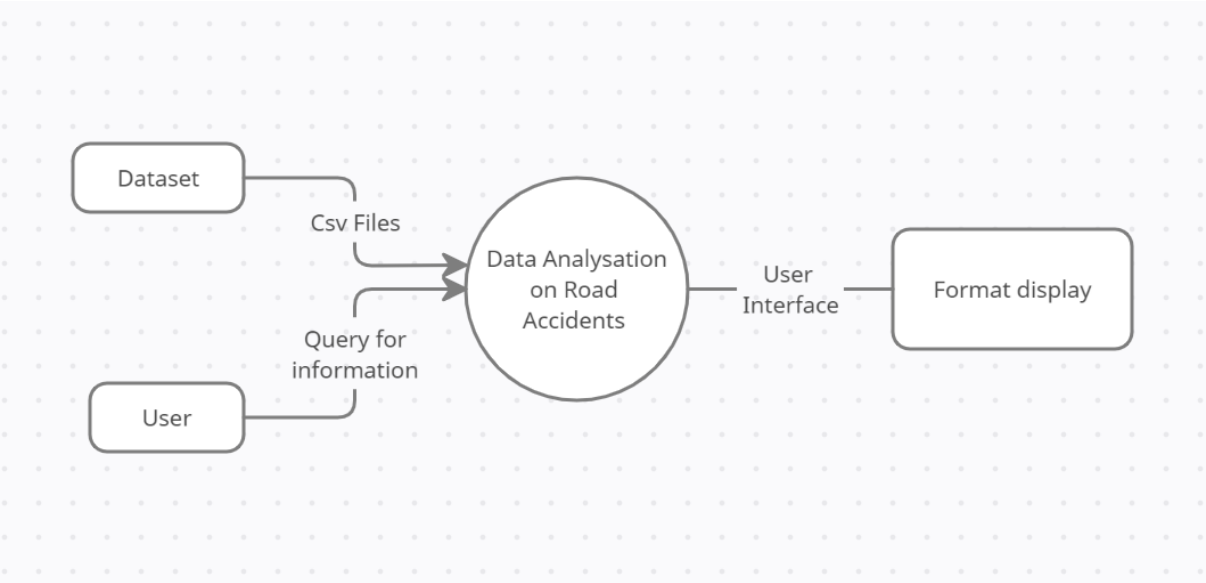


Fig. Level 0 DFD (Context Level DFD)

3.1.2 **Level 1 DFD**

A level 1 DFD notates each of the main sub-processes that together form the complete system. We can think of a level 1 DFD as an “exploded view” of the context diagram.

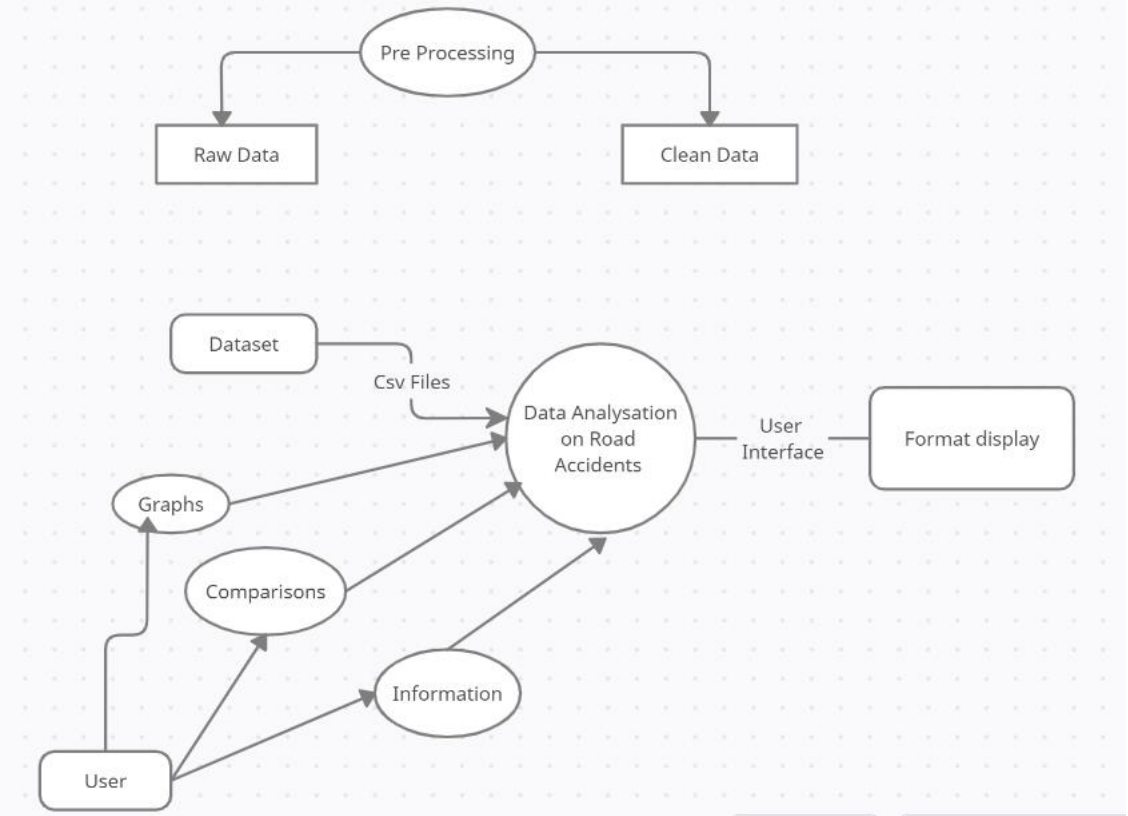


Fig. Level 1 DFD

3.1.3 **Level 2 DFD**

A level 2 data flow diagram (DFD) offers a more detailed look at the processes that make up an information system than a level 1 DFD does. It can be used to plan or record the specific makeup of a system. You can then input the particulars of your own system.

3.2 **Data Dictionary**

A data dictionary is a file or a set of files that includes a database's metadata. The data dictionary holds records about other objects in the database, such as data ownership, data relationships to other objects, and other data. The data dictionary is an essential component of any relational database.

Field Name	Data Type	Data Format	Field Size	Description
Name	String	[A-Z][a-z]	20	Name of user
Email Id	String	[A-Z|a-z|0-9]*[@+[.]	30	Email-Id of user
Username	String	[A-Z][a-z]	20	Username for login
Password	String	[A-Z|a-z|0-9]*[@+[$]	20	Password for login

3.3 **Use Case Diagram**

A use case diagram is a dynamic or behavior diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform. ... The "actors" are people or entities operating under defined roles within the system.

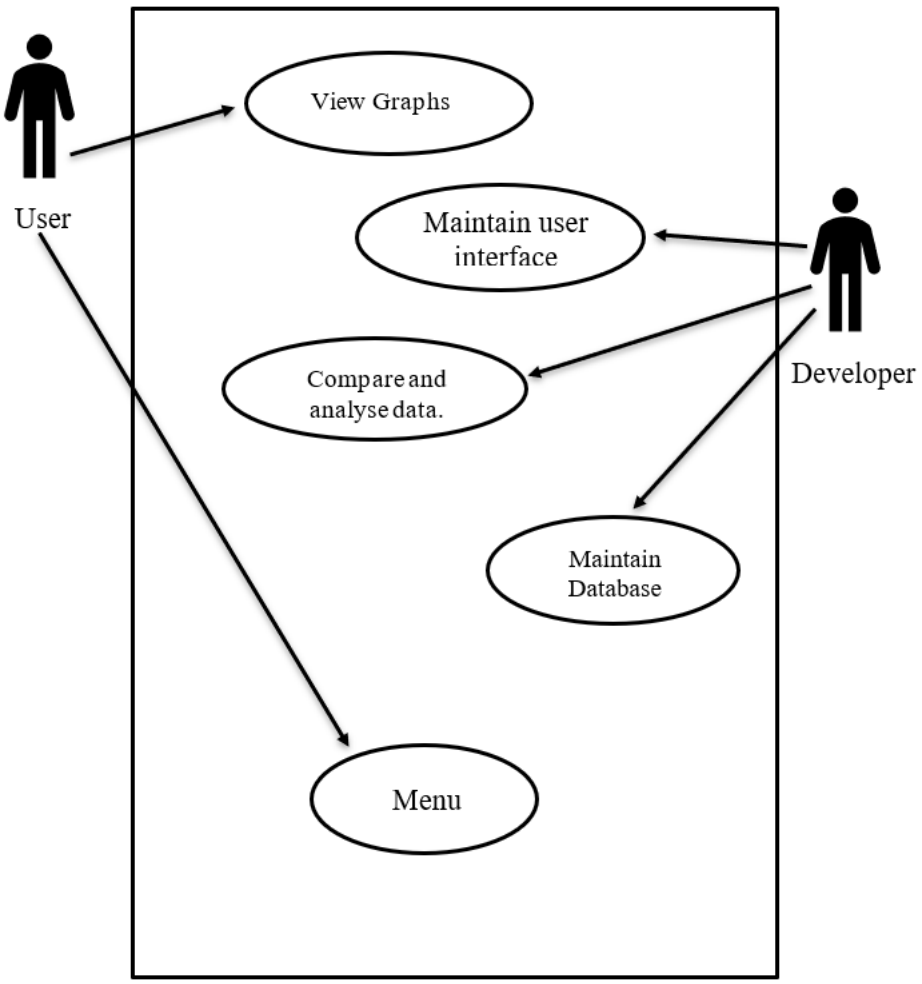


Fig. Use case Diagram for Data Analysis

3.4 **Sequence Diagram**

A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process.

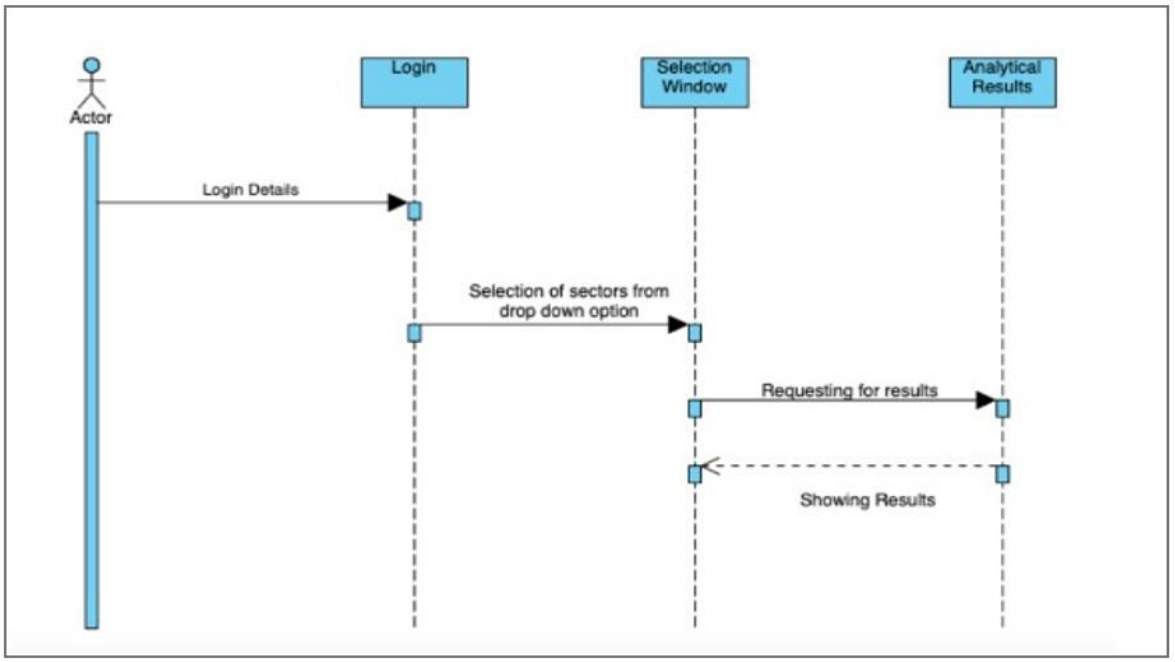


Fig. Sequence Diagram

Chapter 4

SOFTWARE REQUIREMENT SPECIFICATION

A Software Requirement Specification (SRS) is a comprehensive description of the intended purpose and environment for software under development. The SRS fully describes what the software will do and how it will be expected to perform.

4.1 Overall Description

The purpose of the Data analysing of road accidents in India is to clearly define the analyzation under development, namely 'Data Analysis Road Accidents'. The intended audience of this document includes the development team such as the requirements team, requirements analyst, design team, and other members of the developing Organisation.

4.1.1 Product Function

Our project is on data analysis of road accidents using python. This software simplifies the functioning of analyzation and the no. of possibilities of accidents in the country or differents parts of country or different zones.

4.1.2 User Characteristics

There are 2 main groups of this analyzation: i. Users ii. Volunteers

4.1.3 General constraints

- ✓ This system provides access to all users and our volunteers.
- ✓ The user interface will be intuitive enough so that no training is required by volunteers and users.
- ✓ Persistent storage for data sets and for all user information will be maintained.

4.1.4 Assumptions and Dependencies

- ❖ It is assumed that the user is familiar with an internet browser and also familiar in handling a keyboard and mouse or a phone.
- ❖ It is assumed that road accident analyzation of different states is divided into different zones. Each zone have its specific no. of states.
- ❖ It is assumed that analyzation of road accidents is also done on the basis of weather conditions and road accidents.
- ❖ It is assumed that the maintenance work be available sometime after completion of the project.

4.2 External Interface Requirements

4.2.1 User Interface

The user interface is supposed to be design in a very simple and easy to use. The user can directly check the analyzation of accidents of a specific area or a location or a zone.

4.2.2 Hardware Interface

No such hardware interface is required, it just takes user information and stores it into the database.

4.2.3 Software Interface

- Working on different datasets using python and python libraries.
- To store all data on the databases.
- Using SQL for storing data in databases.
- No need of any external software interaction.

4.3 Functional Requirements

4.3.1 FR1 Login Requirement

4.3.1.1 Log In

The volunteers will enter their login id and password to gain access to the portal.

4.3.1.2 Forgot Password

A link will be sent to the registered email account to reset the password.

4.3.1.3 Change Password

A link will be sent to the registered email account to change the password.

4.3.2 FR2 Registration Form Requirement

4.3.2.1 Registration

The volunteers will first register themselves through the registration portal and create their account. Volunteers need to submit some of their basic details like as Name, Email, Phone no, DOB …etc.

Code Snippets

Code snippet 1 (import libraries)

```
In [35]: import pandas as pd
         import numpy as np
         import seaborn as sns
         import matplotlib.pyplot as plt
```

Code snippet 2 (Reading Data)

Reading Data

```
In [5]: killed_df =  pd.read_csv("Road_Accidents_2017-Annuxure_Tables_3.csv")
        injured_df =  pd.read_csv("Road_Accidents_2017-Annuxure_Tables_4.csv")

        weather_df =  pd.read_csv("Acc_Classified_according_to_Type_of_Weather_Condition_2014_and_2016.csv")
        roadcond_df =  pd.read_csv("Acc_clf_acco_to_Road_Cond_2014_and_2016.csv")
```

Code snippet 3 (Data Clean up)

Data Cleanup

In [6]:
```
killed_df = killed_df.drop(columns = ['Share of States/UTs in Total Number of Persons Killed in Road Accidents - 2014',
       'Share of States/UTs in Total Number of Persons Killed in Road Accidents - 2015',
       'Share of States/UTs in Total Number of Persons Killed in Road Accidents - 2016',
       'Share of States/UTs in Total Number of Persons Killed in Road Accidents - 2017',
       'Total Number of Persons Killed in Road Accidents Per Lakh Population - 2014',
       'Total Number of Persons Killed in Road Accidents Per Lakh Population - 2015',
       'Total Number of Persons Killed in Road Accidents Per Lakh Population - 2016',
       'Total Number of Persons Killed in Road Accidents Per Lakh Population - 2017',
       'Total Number of Persons Killed in Road Accidents per 10,000 Vehicles - 2014',
       'Total Number of Persons Killed in Road Accidents per 10,000 Vehicles - 2015',
       'Total Number of Persons Killed in Road Accidents per 10,000 Vehicles - 2016',
       'Total Number of Persons Killed in Road Accidents per 10,000 Km of Roads - 2014',
       'Total Number of Persons Killed in Road Accidents per 10,000 Km of Roads - 2015',
       'Total Number of Persons Killed in Road Accidents per 10,000 Km of Roads - 2016'])
injured_df = injured_df.drop(columns = ['Share of States/UTs in Total Number of Persons Injured in Road Accidents - 2014',
       'Share of States/UTs in Total Number of Persons Injured in Road Accidents - 2015',
       'Share of States/UTs in Total Number of Persons Injured in Road Accidents - 2016',
       'Share of States/UTs in Total Number of Persons Injured in Road Accidents - 2017',
       'Total Number of Persons Injured in Road Accidents Per Lakh Population - 2014',
       'Total Number of Persons Injured in Road Accidents Per Lakh Population - 2015',
       'Total Number of Persons Injured in Road Accidents Per Lakh Population - 2016',
       'Total Number of Persons Injured in Road Accidents Per Lakh Population - 2017',
       'Total Number of Persons injured in Road Accidents per 10,000 Vehicles - 2014',
       'Total Number of Persons injured in Road Accidents per 10,000 Vehicles - 2015',
       'Total Number of Persons injured in Road Accidents per 10,000 Vehicles - 2016',
       'Total Number of Persons injured in Road Accidents per 10,000 Km of Roads - 2014',
       'Total Number of Persons injured in Road Accidents per 10,000 Km of Roads - 2015',
       'Total Number of Persons injured in Road Accidents per 10,000 Km of Roads - 2016'])
```

Code snippet 4 (Dividing states into zones)

Dividing States into Zones and Adding a Column

In [8]:
```
north_india = ['Jammu & Kashmir', 'Punjab', 'Himachal Pradesh', 'Haryana', 'Uttarakhand', 'Uttar Pradesh', 'Chandigarh', 'Jammu a
east_india = ['Bihar', 'Odisha', 'Jharkhand', 'West Bengal', 'Orissa']
south_india = ['Andhra Pradesh', 'Karnataka', 'Kerala' ,'Tamil Nadu', 'Telangana']
west_india = ['Rajasthan' , 'Gujarat', 'Goa','Maharashtra','Goa']
central_india = ['Madhya Pradesh', 'Chhattisgarh']
north_east_india = ['Assam', 'Sikkim', 'Nagaland', 'Meghalaya', 'Manipur', 'Mizoram', 'Tripura', 'Arunachal Pradesh']
ut_india = ['Andaman and Nicobar Islands', 'Dadra and Nagar Haveli', 'Puducherry', 'Andaman & Nicobar Islands', 'Dadra & Nagar Ha
```

In [9]:
```
def get_zonal_names(row):
    if row['States/UTs'].strip() in north_india:
        val = 'North Zone'
    elif row['States/UTs'].strip()  in south_india:
        val = 'South Zone'
    elif row['States/UTs'].strip()  in east_india:
        val = 'East Zone'
    elif row['States/UTs'].strip()  in west_india:
        val = 'West Zone'
    elif row['States/UTs'].strip()  in central_india:
        val = 'Central Zone'
    elif row['States/UTs'].strip()  in north_east_india:
        val = 'NE Zone'
    elif row['States/UTs'].strip()  in ut_india:
        val = 'Union Terr'
    else:
        val = 'No Value'
    return val
```

Code snippet 5 (Graph of various data analyzation)

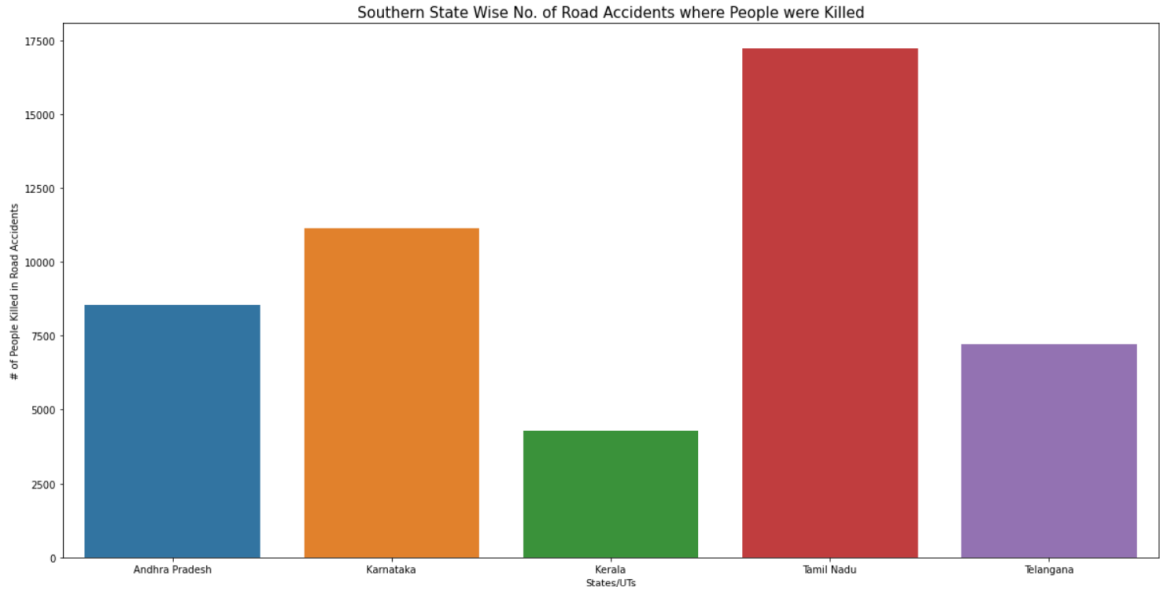
No. of accidents where people were killed in southern zone states:-

Southern-Zone States: No. of Road Accidents where People were Killed

```
In [15]: sub_df = killed_df[killed_df['Zones'] == 'South Zone']
         df =pd.pivot_table(sub_df, index=['States/UTs'],values=[2014, 2015, 2016, 2017],aggfunc=np.sum).reset_index()
         df

         years = [2014,2015,2016,2017]
         fig,ax = plt.subplots(1,1, figsize=(20,10))
         for i,year in enumerate(years):
             sns.barplot(x=df['States/UTs'],y=df[year])
             plt.ylabel('# of People Killed in Road Accidents')
             plt.title('Southern State Wise No. of Road Accidents where People were Killed', fontsize=15)
```

Output:-



Weather Conditions - No. of People Injured in Road Accidents (South Zone)

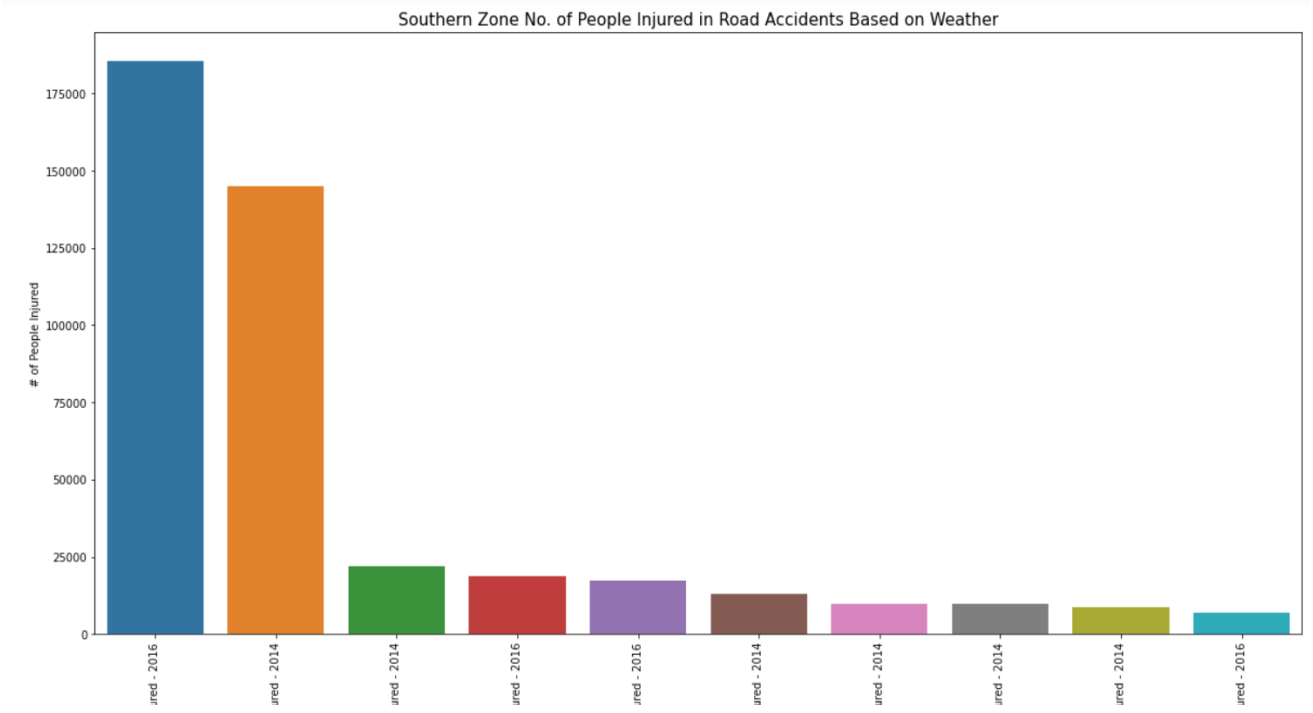
In [19]:
```
sub_df = weather_df_injured[weather_df_injured['Zones'] == 'South Zone']
df =pd.pivot_table(sub_df, index=['Zones'],aggfunc=np.sum).reset_index()
df = df.T.reset_index()
df = df.rename(columns = {'index': 'Weather Conditions', 0: 'Total'})
df = df.drop(df.index[0])
df = df.sort_values(by = ['Total'], ascending=False).head(10)
df

fig,ax = plt.subplots(1,1, figsize=(20,10))

sns.barplot(x=df['Weather Conditions'],y=df['Total'])
plt.ylabel('# of People Injured')
plt.title('Southern Zone No. of People Injured in Road Accidents Based on Weather', fontsize=15)
plt.xticks(rotation=90)
```

Out[19]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(0, 0, 'Fine/Clear - Persons Injured - 2016'),
  Text(1, 0, 'Fine - Persons Injured - 2014'),
  Text(2, 0, 'Other extraordinary weather condition - Persons Injured - 2014'),
  Text(3, 0, 'Others - Persons Injured - 2016'),
  Text(4, 0, 'Rainy - Persons Injured - 2016'),
  Text(5, 0, 'Light rain - Persons Injured - 2014'),
  Text(6, 0, 'Very hot - Persons Injured - 2014'),
  Text(7, 0, 'Heavy rain - Persons Injured - 2014'),
  Text(8, 0, 'Cloudy - Persons Injured - 2014'),
  Text(9, 0, 'Cloudy - Persons Injured - 2016')])
```

Output:-



Road Conditions - No. of People Injured in Road Accidents (South Zone)

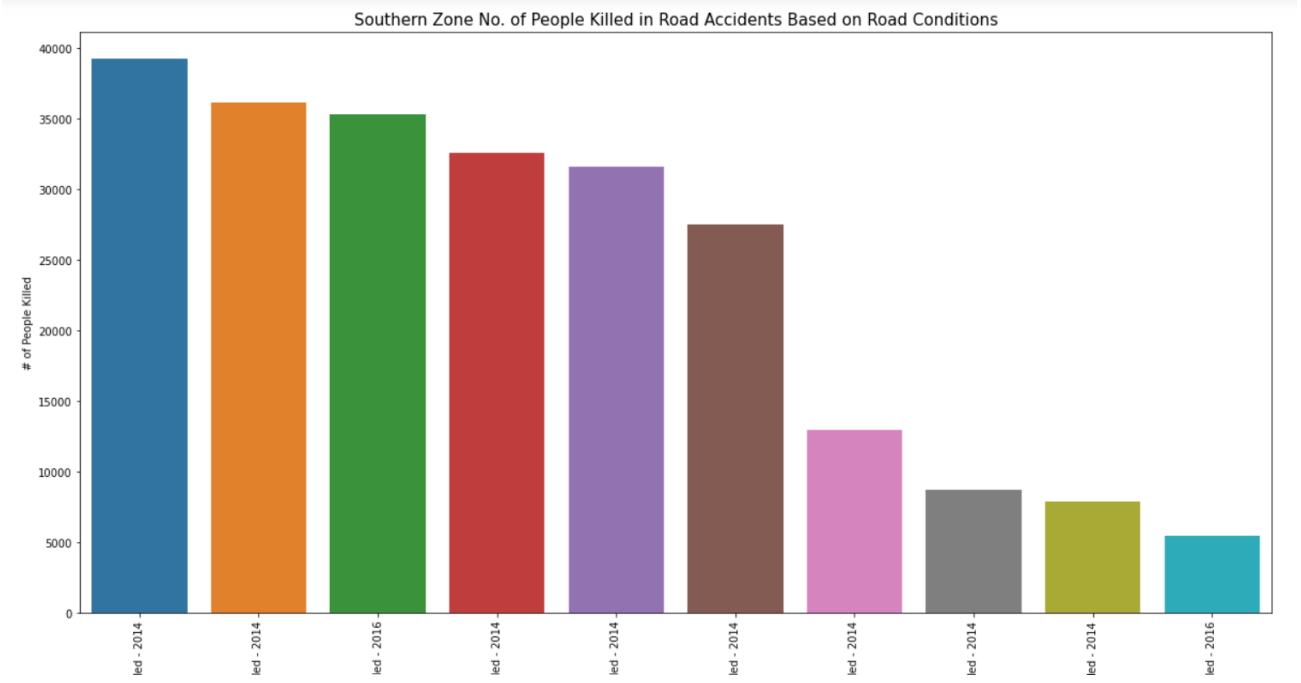
Road Conditions - No. of People Killed in Road Accidents (South Zone)

In [21]:
```
sub_df = roadcond_df_killed[roadcond_df_killed['Zones'] == 'South Zone']
df =pd.pivot_table(sub_df, index=['Zones'],aggfunc=np.sum).reset_index()
df = df.T.reset_index()
df = df.rename(columns = {'index': 'Road Conditions', 0: 'Total'})
df = df.drop(df.index[0])
df = df.sort_values(by = ['Total'], ascending=False).head(10)
df

fig,ax = plt.subplots(1,1, figsize=(20,10))
sns.barplot(x=df['Road Conditions'],y=df['Total'])
plt.ylabel('# of People Killed')
plt.title('Southern Zone No. of People Killed in Road Accidents Based on Road Conditions', fontsize=15)
plt.xticks(rotation=90)
```

Out[21]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(0, 0, 'Dry road- Killed - 2014'),
  Text(1, 0, 'Flat Road-Killed - 2014'),
  Text(2, 0, 'Pucca road (Normal Road) - Persons Killed - 2016'),
  Text(3, 0, 'Straight Road-Killed - 2014'),
  Text(4, 0, 'Surfaced Roads- Killed - 2014'),
  Text(5, 0, 'Good surface- Killed - 2014'),
  Text(6, 0, 'Others- Killed - 2014'),
  Text(7, 0, 'Metalled Roads- Killed - 2014'),
  Text(8, 0, 'Slight Curve-Killed - 2014'),
  Text(9, 0, 'Others - Persons Killed - 2016')])
```

Output:-



Weather Conditions - No. of People Killed in Road Accidents (North Zone)

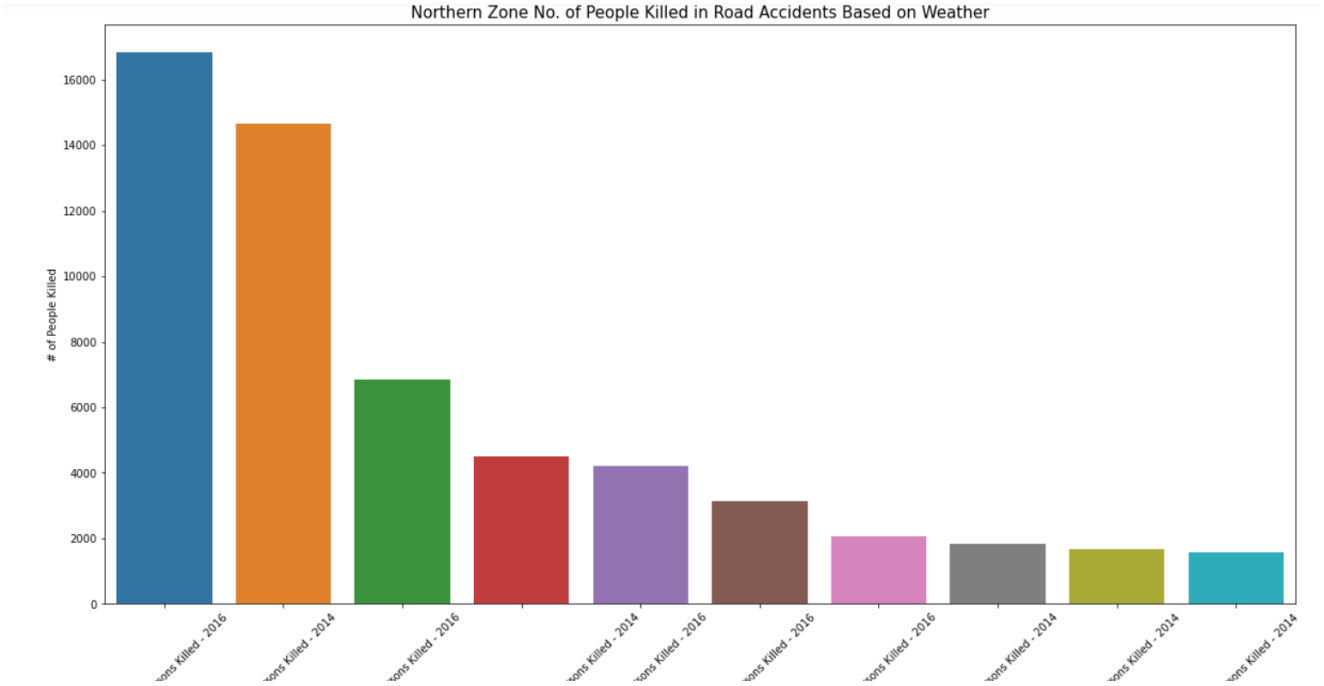
In [38]:

In [38]:
```
sub_df = weather_df_killed[weather_df_killed['Zones'] == 'North Zone']
df =pd.pivot_table(sub_df, index=['Zones'],aggfunc=np.sum).reset_index()
df = df.T.reset_index()
df = df.rename(columns = {'index': 'Weather Conditions', 0: 'Total'})
df = df.drop(df.index[0])
df = df.sort_values(by = ['Total'], ascending=False).head(10)
df

fig,ax = plt.subplots(1,1, figsize=(20,10))
sns.barplot(x=df['Weather Conditions'],y=df['Total'])
plt.ylabel('# of People Killed')
plt.title('Northern Zone No. of People Killed in Road Accidents Based on Weather', fontsize=15)
plt.xticks(rotation=45)
```

Out[38]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(0, 0, 'Fine/Clear - Persons Killed - 2016'),
  Text(1, 0, 'Fine - Persons Killed - 2014'),
  Text(2, 0, 'Others - Persons Killed - 2016'),
  Text(3, 0, 'Other extraordinary weather condition - Persons Killed - 2014'),
  Text(4, 0, 'Mist/ Foggy - Persons Killed - 2016'),
  Text(5, 0, 'Rainy - Persons Killed - 2016'),
  Text(6, 0, 'Cloudy - Persons Killed - 2016'),
  Text(7, 0, 'Mist/fog - Persons Killed - 2014'),
  Text(8, 0, 'Light rain - Persons Killed - 2014'),
  Text(9, 0, 'Very cold - Persons Killed - 2014')])
```

Output:-



1.2 Pseudo Code

It's simply an implementation of an algorithm in the form of annotations and informative text written in plain English. It has no syntax like any of the programming language and thus can't be compiled or interpreted by the computer.

Module 1 : Login
Input : First name, last name, login id
Output : directed to next

Begin

1. Create a class
2. Make a Tkinter window object
3. Set widgets like frame, labels, entry, image and button to the window.
4. Login button will execute the check function.
5. Exit button will destroy the window.

Function to check whether the entry fields are empty or not

```
Check()
if self.fname.get()=="" or self.lname.get()=="" or self.id.get()=="" :
    print("Error","All fields are required")
else
    print("Welcome to DeathIN")
```

End

Module 2 : Data Analysis
Input : CSV files
Output : graphs

1. Begin.
2. Import required modules of python.
3. Data = reading the CSV file and storing the dataframe.
4. Data cleaning of dataframe using head(), tail() or iloc[] function.
5. Plot the desired graphs for the dataset.
6. Find the data of road accidents of India from the dataset.
7. Analyzation of data from 2014 to 2017
8. Making a Tkinter window.

9. Placing a button “Show Graph”.
10. Show Graph will display a graph.
11. btn = Button(window, text = 'Show Graph', command = graph)
12. End.