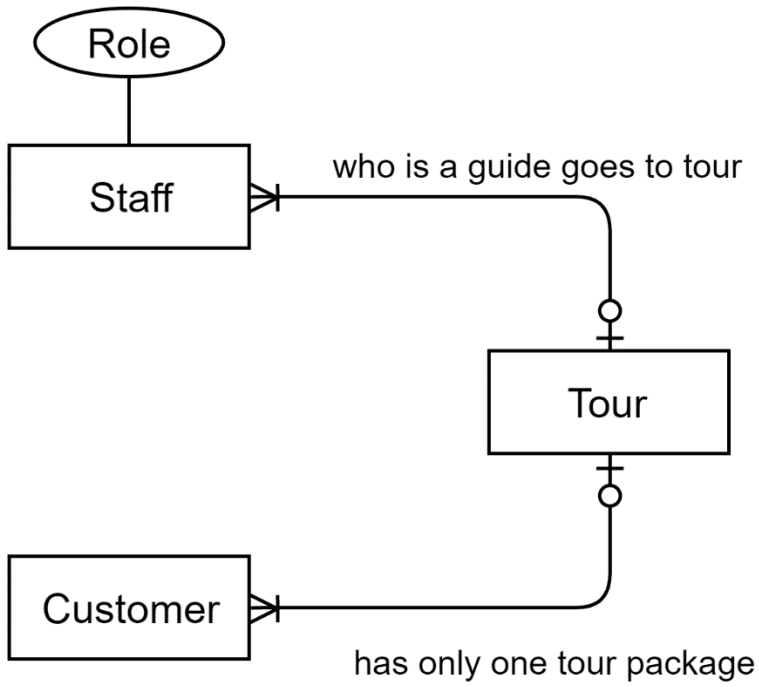
# Textual Analysis

According to the scenario the tour company have the following relation between the entities:

1. Staffs have different roles assigned one for each
2. Multiple Staffs (specifically tour guides) could be assigned for a tour
3. Customer can only take one package from multiple available packages.
4. The tour and package are taken as a same entity
5. A registered customer might or might not have a tour assigned
6. Role is taken as an attribute of staff



# UNF

Storing the data given in unique rows in a table we can observe:

| Package ID | Package Name | Destination | Total No of Days | Difficulty |
|---|---|---|---|---|
| LK25A | ABC | Annapurna Base Camp | 7 | Moderate |
| UI32A | Ghandruk | Ghandruk | 4 | Moderate |
| UI32A | Ghandruk | Pokhara | 4 | Moderate |
| NB34G | Everest Short Trek | Khumjung | 4 | Hard |
| NB34G | Everest Short Trek | Lukla | 4 | Hard |

From the above table we can observe that the destination column has repeating groups in it:

UNF: (**Package_ID**, Pakage_Name, {Destination}, Total_Days, Difficulty)

Destination is the repeating group because a package might have multiple destinations for example Pokhara and Ghandruk. Package_ID is taken as the primary key as it can uniquely identify all columns.

# 1NF

As Pakage_Name, Total_Days, and Difficulty depends upon the Package_ID, it is assigned as a primary key.

Package_ID ➔ Pakage_Name, Total_Days, Difficulty

Removing the repeating group and creating the Destination Package and assigning package_ID and Destination_ID as composite key and creating the first normal form.

Package (**Package_ID**, Pakage_Name, Total_Days, Difficulty)
Destination_Package (**Package_ID*, Destination**)

# 2NF

There are no Partial dependencies as all of the non key elements are directly dependent on their respective primary key

Package_ID ➔ Pakage_Name, Total_Days, Difficulty
Package_ID, Destination gives the unique value for every package-destination pair

Package (**Package_ID**, Pakage_Name, Total_Days, Difficulty)
Destination_Package (**Package_ID***, **Destination**)

# 3NF

There are no transitive dependencies as every none key entries are dependent on the primary key assigned for each table.

Package (**Package_ID**, Pakage_Name, Total_Days, Difficulty)
Destination_Package (**Package_ID***, **Destination**)

Since there are no transative dependencies this is the final normalized form

# UNF

Adding Package ID and storing the data given in unique rows in a table we can observe:

| Pack-age ID | Package Name | Start Date | End Date | Tour Guide | Day | Travel Details | Activities | Status | Travel Mode | Dfficulty |
|---|---|---|---|---|---|---|---|---|---|---|
| UI32A | Ghandruk | 1st Jan 2019 | 7th Jan 2019 | Will Stark | Day 1 | Kathmandu to Pokhara | Driving from KTM to Pokhara Overnight stay in Hotel | Complete | Bus | Easy |
| UI32A | Ghandruk | 1st Jan 2019 | 7th Jan 2019 | Will Stark | Day 2 | Pokhara to Ghandruk | Trek to Ghandruk. Explore the Ghandruk Village. | Complete | Bus/Walk | Hard |
| UI32A | Ghandruk | 1st Jan 2019 | 7th Jan 2019 | Will Stark | Day 3 | Ghandruk to Pokhara | View the beautiful sunrise and Himalayas. Trek Down to Pokhara | Complete | Bus/Walk | Hard |
| UI32A | Ghandruk | 1st Jan 2019 | 7th Jan 2019 | Will Stark | Day 4 | Pokhara to Kathmandu | Drive back to Pokhara | Remaining | Bus | Moderate |

From the above table we can observe that the Day, Travel Details, Activities, Status, travel Mode and Difficuly are the repeating groups:

UNF: (**Package_ID**, Package_Name, Start_Date, End_Date, Tour_Guide, {Day, Travel_Details, Activities, Status, Travel_Mode, Difficulty_Level})

Now separating the Repeating group to a new table in 1NF.

# 1NF

Package ID is taken as the Primary key as it gives

Package_ID ➔ Package_Name, Start_Date, End_Date, Tour_Guide

Removing the repeating group and creating the Itenary_Tour. Assigning package_ID as Primary key and creating the first normal form.

Tour (**Package_ID**, Package_Name, Start_Date, End_Date, Tour_Guide)
Itenary_Tour (**Package_ID***, Day, Travel_Details, Activities, Status, Travel_Mode, Difficulty_Level)

# 2NF

There are no Partial dependencies as all of the non key elements are directly dependent on their respective primary key

Package_ID ➔ Pakage_Name, Total_Days, Difficulty
Package_ID, Destination gives the unique value for every package-destination pair

Package (**Package_ID**, Pakage_Name, Total_Days, Difficulty)
Destination_Package (**Package_ID***, **Destination**)

# 3NF

There are no transitive dependencies as every none key entries are dependent on the primary key assigned for each table.

Package (**Package_ID**, Pakage_Name, Total_Days, Difficulty)
Destination_Package (**Package_ID***, **Destination**)

Since there are no transative dependencies this is the final normalized form