Exploratory Data Analysis on Pradhan Mantri Fasal Bima Yojana(PMFBY) and Weather Based Crop Insurance Scheme(WBCIS)

The Pradhan Mantri Fasal Bima Yojana was launched in 2016 with a aim to provide insurance coverage and financial support to the farmers in the event of failure of any of the notified crops, from pre-sowing to post-harvest period, as a result of natural calamities, pests, and diseases. The objectives of the scheme also include stabilization of farmers income in order to ensure that they continue farming and to encourage farmers to adopt innovative and modern agricultural practices.

The Weather based crop insurance scheme was launched in 2007 and then it was restructered in 2016. Under WBCIS scheme, farmers are provided with insurance protection against adverse weather incidents such as deficit or excess rainfall, temperature difference, moisture and others which have a significant impact on crop production.

The dataset is taken from kaggle and it has many features like state, year, gross premium etc. The objective is to explore and visualize the dataset using different python packages, so that we could understand the preferences of different types of farmers among both the schemes.

Downloading the Dataset

Firstly, we will download the dataset from the kaggle.

```
!pip install jovian opendatasets --upgrade --quiet
```

Let's begin by downloading the data, and listing the files within the dataset.

```
data_url ="https://www.kaggle.com/datasets/pyatakov/india-pmfby-statistics"
```

```
import opendatasets as od
od.download(data_url)
```

```
Please provide your Kaggle credentials to download this dataset. Learn more:
http://bit.ly/kaggle-creds
Your Kaggle username: akshitsaxena20
Your Kaggle Key: ········
Downloading india-pmfby-statistics.zip to ./india-pmfby-statistics

100%|██████████| 1.23M/1.23M [00:00<00:00, 72.9MB/s]
```

The dataset has been downloaded and extracted.

```
data_dir = './india-pmfby-statistics'
```

```
import os
os.listdir(data_dir)
```

```
['PMFBY statistics.csv', 'PMFBY coverage.csv']
```

Let us save and upload our work to Jovian before continuing.

```
project_name = "Exploratory Data Analysis on Pradhan Mantri Fasal Bima Yojana(PMFBY) ar
```

```
!pip install jovian --upgrade -q
```

```
import jovian
```

```
jovian.commit(project=project_name)
```

```
[jovian] Creating a new project "saxenaakshit123/Exploratory Data Analysis on Pradhan mantri fasal bima yojana(PMFBY) and Weather based crop insurance scheme(WBCIS)"
[jovian] Committed successfully! https://jovian.ai/saxenaakshit123/exploratory-data-analysis-on-pradhan-mantri-fasal-bima-yojana-pmfby-and-weather-based-crop-d7271
```

```
'https://jovian.ai/saxenaakshit123/exploratory-data-analysis-on-pradhan-mantri-fasal-bima-yojana-pmfby-and-weather-based-crop-d7271'
```

Data Preparation and Cleaning

First we will read the data file and then we will prepare and clean the data to proceed for the further analysis.

```
import numpy as np
import pandas as pd
```

```
df=pd.read_csv(data_dir+ '/PMFBY statistics.csv')
df.info()
df.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5058 entries, 0 to 5057
Data columns (total 29 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   year                           5058 non-null   int64
 1   season                         5058 non-null   object
 2   scheme                         5058 non-null   object
 3   state                          5058 non-null   object
 4   district.farmerCount           5058 non-null   int64
 5   district.application.loanee    5057 non-null   float64
 6   district.nonLoanee             5058 non-null   int64
```

```
 7   district.areaInsured (th.ha)     5058 non-null   float64
 8   district.sumInsured (lac.)       5058 non-null   float64
 9   district.farmerShare (lac.)      5058 non-null   float64
 10  district.goiShare (lac.)         5058 non-null   float64
 11  district.stateShare (lac.)       5058 non-null   float64
 12  district.gender.male (%)         4988 non-null   float64
 13  district.gender.female (%)       4988 non-null   float64
 14  district.gender.transgender (%)  4988 non-null   float64
 15  district.category.sc (%)         4988 non-null   float64
 16  district.category.st (%)         4988 non-null   float64
 17  district.category.obc (%)        4988 non-null   float64
 18  district.category.gen (%)        4988 non-null   float64
 19  district.type.marginal (%)       4988 non-null   float64
 20  district.type.small (%)          4988 non-null   float64
 21  district.type.other (%)          4988 non-null   float64
 22  district.iuCount                 5058 non-null   int64
 23  district.updatedAt               5058 non-null   object
 24  district.districtName            5058 non-null   object
 25  district.districtCode            5058 non-null   int64
 26  district.scheme                  5058 non-null   int64
 27  district.grossPremium (lac.)     5058 non-null   float64
 28  district.nextLevelParams         5058 non-null   object
dtypes: float64(17), int64(6), object(6)
memory usage: 1.1+ MB
```

```
(5058, 29)
```

Here,we can observe that there are 5058 rows and 29 columns in the dataframe. We will remove the unwanted columns from the dataframe.

```
df.drop(columns=["district.application.loanee","district.nonLoanee","district.iuCount",
```

And also we will rename the columns for convenience.

```
df.rename(columns = {'district.farmerCount':'farmercount','district.areaInsured (th.ha)
```

```
df.info()
df.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5058 entries, 0 to 5057
Data columns (total 22 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   year                        5058 non-null   int64
```

```
 1   season                      5058 non-null   object
 2   scheme                      5058 non-null   object
 3   state                       5058 non-null   object
 4   farmercount                 5058 non-null   int64
 5   areaInsured_th_ha           5058 non-null   float64
 6   sumInsured_lakh             5058 non-null   float64
 7   farmerShare_lakh            5058 non-null   float64
 8   goiShare_lakh               5058 non-null   float64
 9   stateShare_lakh             5058 non-null   float64
 10  gender_male_percent         4988 non-null   float64
 11  gender_female_percent       4988 non-null   float64
 12  gender_transgender_percent  4988 non-null   float64
 13  category_sc_percent         4988 non-null   float64
 14  category_st_percent         4988 non-null   float64
 15  category_obc_percent        4988 non-null   float64
 16  category_gen_percent        4988 non-null   float64
 17  type_marginal_percent       4988 non-null   float64
 18  type_small_percent          4988 non-null   float64
 19  type_other_percent          4988 non-null   float64
 20  districtName                5058 non-null   object
 21  grossPremium_lakh           5058 non-null   float64
dtypes: float64(16), int64(2), object(4)
memory usage: 869.5+ KB
```

```
(5058, 22)
```

Now, we have 22 columns in our dataframe. Now, we can see that some of the numeric columns have non-null count less than the total number of rows, i.e., 5058, this means that there are empty cells in those columns. We will remove those rows which contains any empty cells.

```
df=df.dropna()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4988 entries, 0 to 5057
Data columns (total 22 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   year                        4988 non-null   int64
 1   season                      4988 non-null   object
 2   scheme                      4988 non-null   object
 3   state                       4988 non-null   object
 4   farmercount                 4988 non-null   int64
 5   areaInsured_th_ha           4988 non-null   float64
 6   sumInsured_lakh             4988 non-null   float64
```

```
 7   farmerShare_lakh           4988 non-null   float64
 8   goiShare_lakh              4988 non-null   float64
 9   stateShare_lakh            4988 non-null   float64
 10  gender_male_percent        4988 non-null   float64
 11  gender_female_percent      4988 non-null   float64
 12  gender_transgender_percent 4988 non-null   float64
 13  category_sc_percent        4988 non-null   float64
 14  category_st_percent        4988 non-null   float64
 15  category_obc_percent       4988 non-null   float64
 16  category_gen_percent       4988 non-null   float64
 17  type_marginal_percent      4988 non-null   float64
 18  type_small_percent         4988 non-null   float64
 19  type_other_percent         4988 non-null   float64
 20  districtName               4988 non-null   object
 21  grossPremium_lakh          4988 non-null   float64
dtypes: float64(16), int64(2), object(4)
memory usage: 896.3+ KB
```

So, this is our final dataset on which we will perform the EDA. Lets look at the first 5 rows of the data.

```
df.head(5)
```

	year	season	scheme	state	farmercount	areaInsured_th_ha	sumInsured_lakh	farmerShare_lakh	goiShare_la
0	2018	Kharif	WBCIS	ANDHRA PRADESH	530381	898.02	427253.53	8589.30	23135.:
1	2018	Kharif	WBCIS	ANDHRA PRADESH	32478	27.45	19028.48	666.72	428.(
2	2018	Kharif	WBCIS	ANDHRA PRADESH	80	0.20	121.44	6.07	1.:
3	2018	Kharif	WBCIS	ANDHRA PRADESH	10077	8.24	11767.03	588.35	199.:
4	2018	Kharif	WBCIS	ANDHRA PRADESH	97	0.12	164.42	8.22	4.:

5 rows × 22 columns

Also , lets divide this dataframe in two dataframes for each scheme-PMFBY and WBCIS.

```
pmfby_df=df[df.scheme=='PMFBY']
wbcis_df=df[df.scheme=='WBCIS']
```

```
import jovian
```

```
jovian.commit()
```

```
[jovian] Updating notebook "saxenaakshit123/exploratory-data-analysis-on-pradhan-mantri-fasal-bima-yojana-pmfby-and-weather-based-crop-d7271" on https://jovian.ai
```

```
[jovian] Committed successfully! https://jovian.ai/saxenaakshit123/exploratory-data-analysis-on-pradhan-mantri-fasal-bima-yojana-pmfby-and-weather-based-crop-d7271
```

```
'https://jovian.ai/saxenaakshit123/exploratory-data-analysis-on-pradhan-mantri-fasal-bima-yojana-pmfby-and-weather-based-crop-d7271'
```

Exploratory Data Analysis and Visualization

Now, we will explore both the dataframes- pmfby_df and wbcis_df using basic statistics and then we will visualize them.

1) Exploring dataframes using basic statistics

```
pmfby_df.describe()
```

	year	farmercount	areaInsured_th_ha	sumInsured_lakh	farmerShare_lakh	goiShare_lakh	stateShare_
count	3556.000000	3.556000e+03	3556.000000	3556.000000	3556.000000	3556.000000	3556.00
mean	2019.389201	3.329664e+04	48.286853	20683.249634	390.690633	1186.024165	1281.79
std	1.115464	5.251466e+04	93.726066	34702.453903	727.561815	3071.842441	3362.30
min	2018.000000	1.000000e+00	0.000000	0.000000	0.000000	0.000000	0.00
25%	2018.000000	2.321000e+03	1.000000	551.285000	3.842500	6.862500	8.18
50%	2019.000000	1.519300e+04	12.735000	6612.255000	93.310000	130.795000	150.36
75%	2020.000000	4.357825e+04	47.580000	22751.442500	405.425000	781.307500	836.70
max	2021.000000	1.061490e+06	1142.050000	275018.700000	7244.420000	33292.160000	40723.02

```
wbcis_df.describe()
```

	year	farmercount	areaInsured_th_ha	sumInsured_lakh	farmerShare_lakh	goiShare_lakh	stateShar
count	1432.000000	1432.000000	1432.000000	1432.000000	1432.000000	1432.000000	1432.0
mean	2019.230447	3896.619413	23.755321	3165.705136	111.503815	227.367103	264.8
std	1.089289	21263.234944	181.838026	20389.614897	400.282540	1145.864637	1422.1
min	2018.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	2018.000000	35.750000	0.020000	8.582500	0.412550	0.259750	0.2
50%	2019.000000	221.000000	0.150000	81.670000	3.970000	5.040000	5.2
75%	2020.000000	1915.000000	2.035000	1172.497500	55.077500	69.225000	78.1
max	2021.000000	530381.000000	3777.400000	535572.500000	8589.300000	23135.370000	32881.6

We can clearly observe that

- count of farmers who enrolled under PMFBY is more than the WBCIS.
- average percentage of marginal and small farmers who has enrolled under PMFBY is more than the average percentage of marginal and small farmers who has enrolled under WBCIS.
- Also, it is obvious that the average sum insured,gross premium and the average shares of farmer,goi and state would be more in PMFBY than WBCIS as the count of farmers is more in PMFBY.

Also, here we can also observe the standard deviation,IQR,minimum and maximum value of each feature.

2) Average number of farmers registered under each scheme per season per year

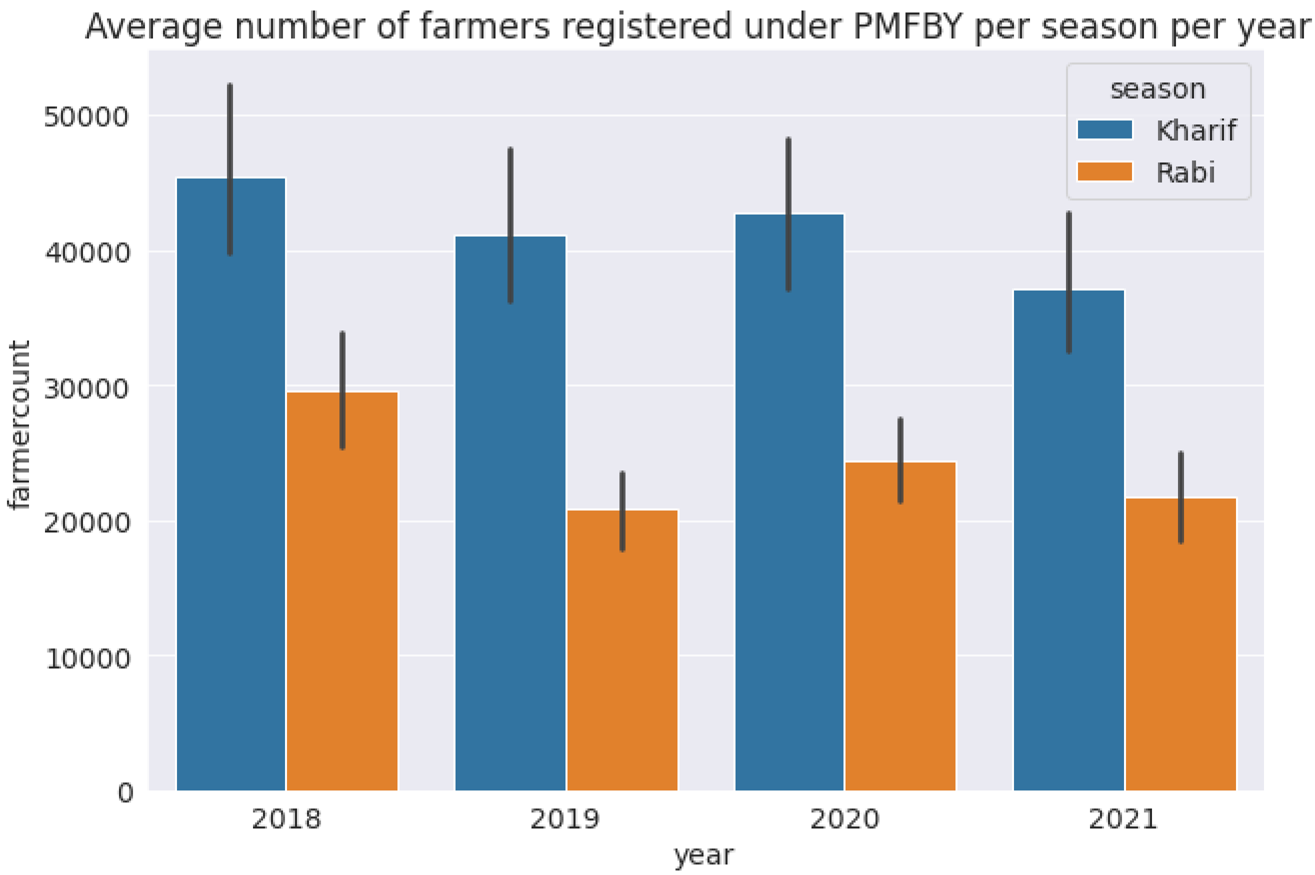
Let's import `matplotlib.pyplot and seaborn` .

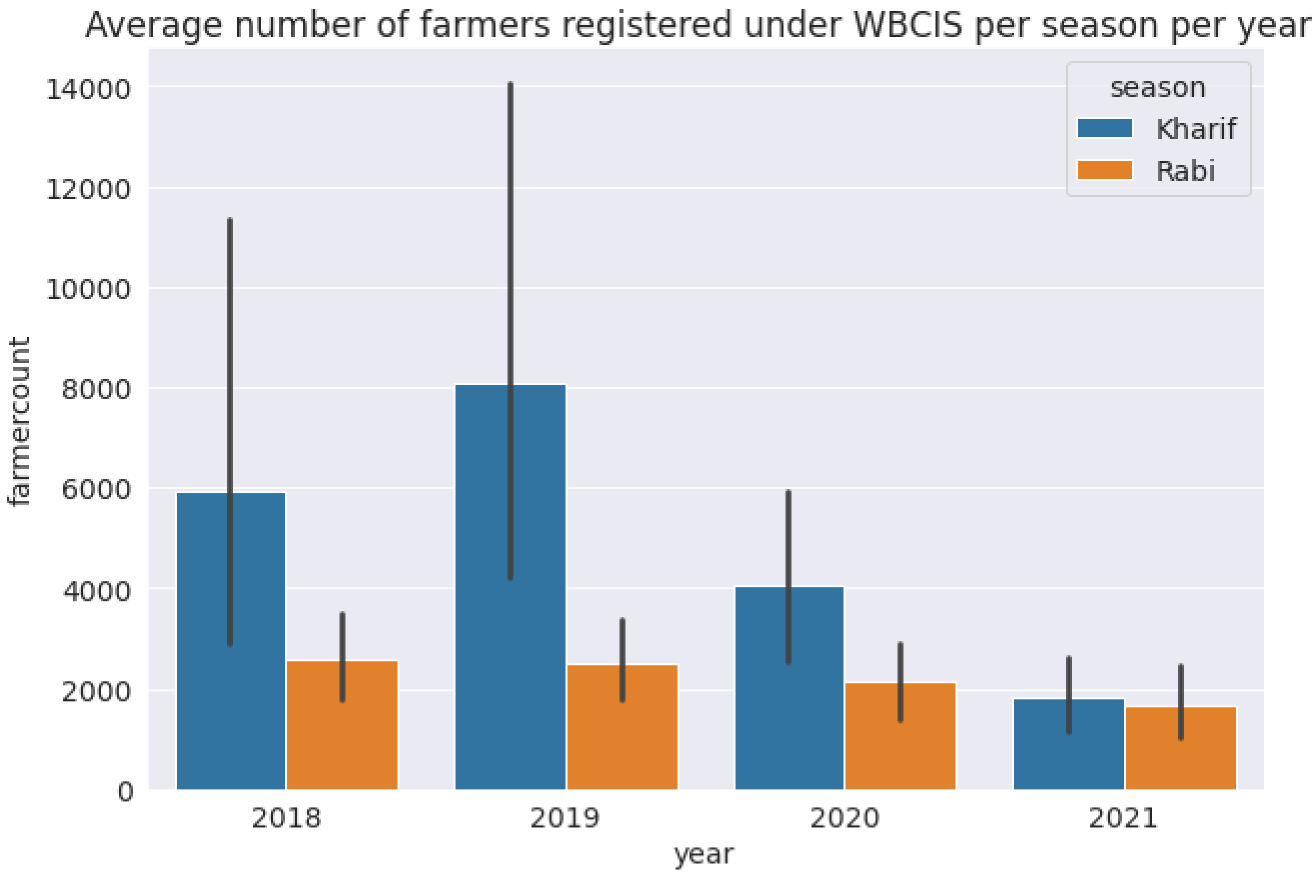
```
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

sns.set_style('darkgrid')
matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (9, 5)
matplotlib.rcParams['figure.facecolor'] = '#00000000'
```

```
fig, ax = plt.subplots(nrows=2,ncols=1,figsize=(10,15))
sns.barplot(x=pmfby_df.year,y=pmfby_df.farmercount,hue=pmfby_df.season,ax=ax[0]).set_ti
sns.barplot(x=wbcis_df.year,y=wbcis_df.farmercount,hue=wbcis_df.season,ax=ax[1]).set_ti
```

```
Text(0.5, 1.0, 'Average number of farmers registered under WBCIS per season per year ')
```



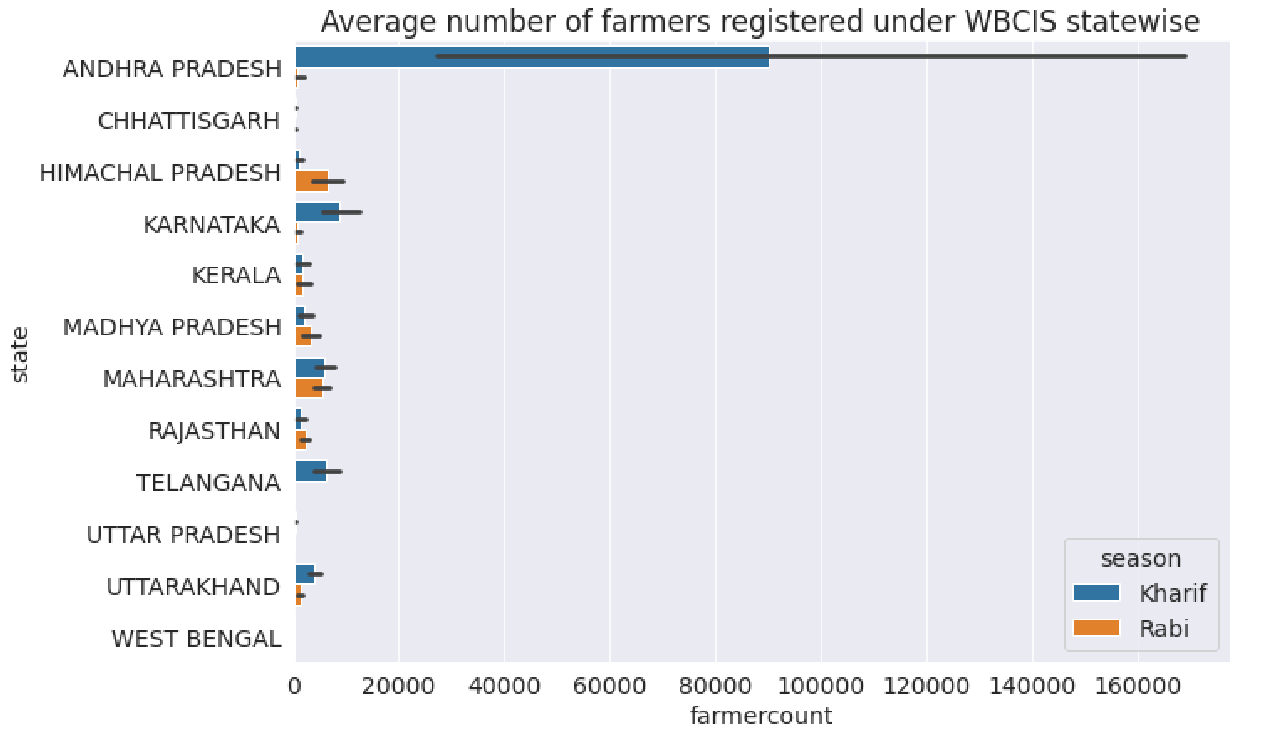


So, as the average number of farmers registered under PMFBY is more than WBCIS ,we can say that most farmers prefer PMFBY more than WBCIS. Even the registrations for WBCIS decreased gradually from 2019. Also, in both schemes, farmers preferred for insurance more during the kharif season.

3) Average number of farmers registered under each scheme statewise

```
fig, ax = plt.subplots(nrows=2,ncols=1,figsize=(10,15))
sns.barplot(y=pmfby_df.state,x=pmfby_df.farmercount,hue=pmfby_df.season,ax=ax[0]).set_t
sns.barplot(y=wbcis_df.state,x=wbcis_df.farmercount,hue=wbcis_df.season,ax=ax[1]).set_t
```

```
Text(0.5, 1.0, 'Average number of farmers registered under WBCIS statewise')
```

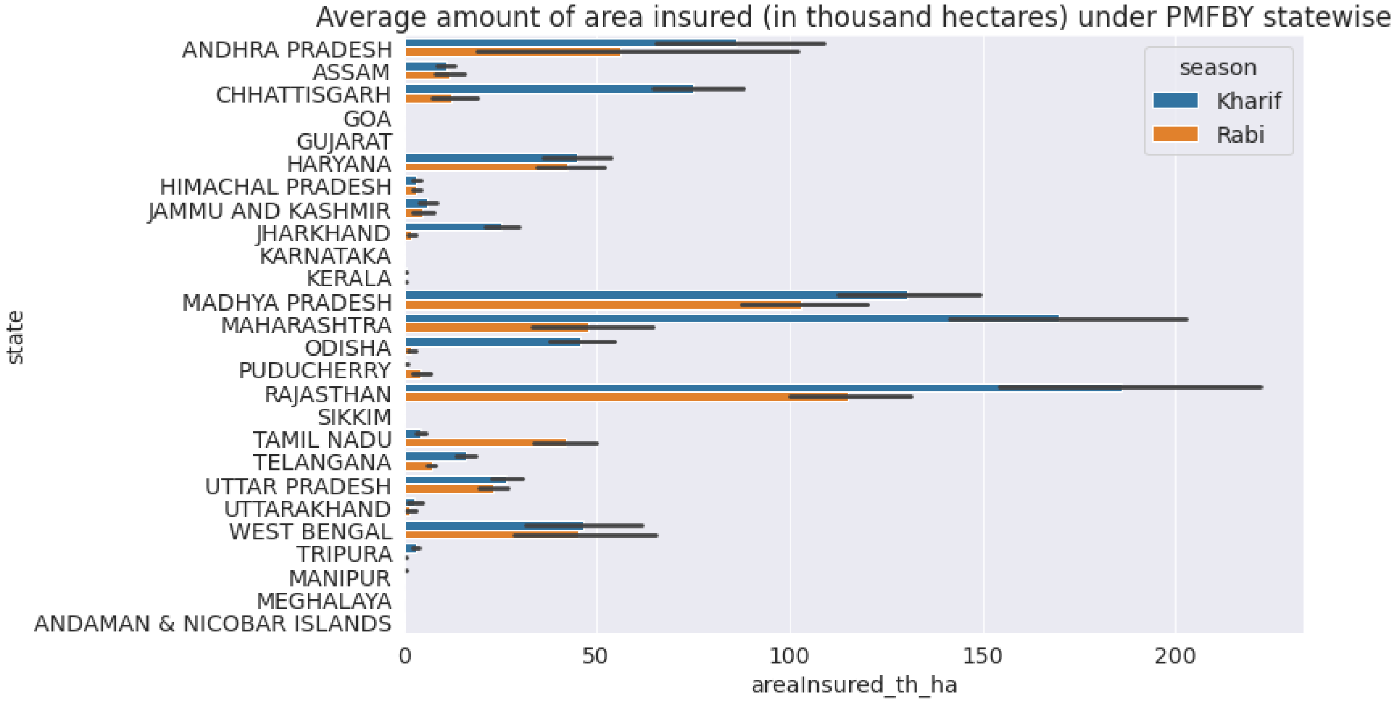


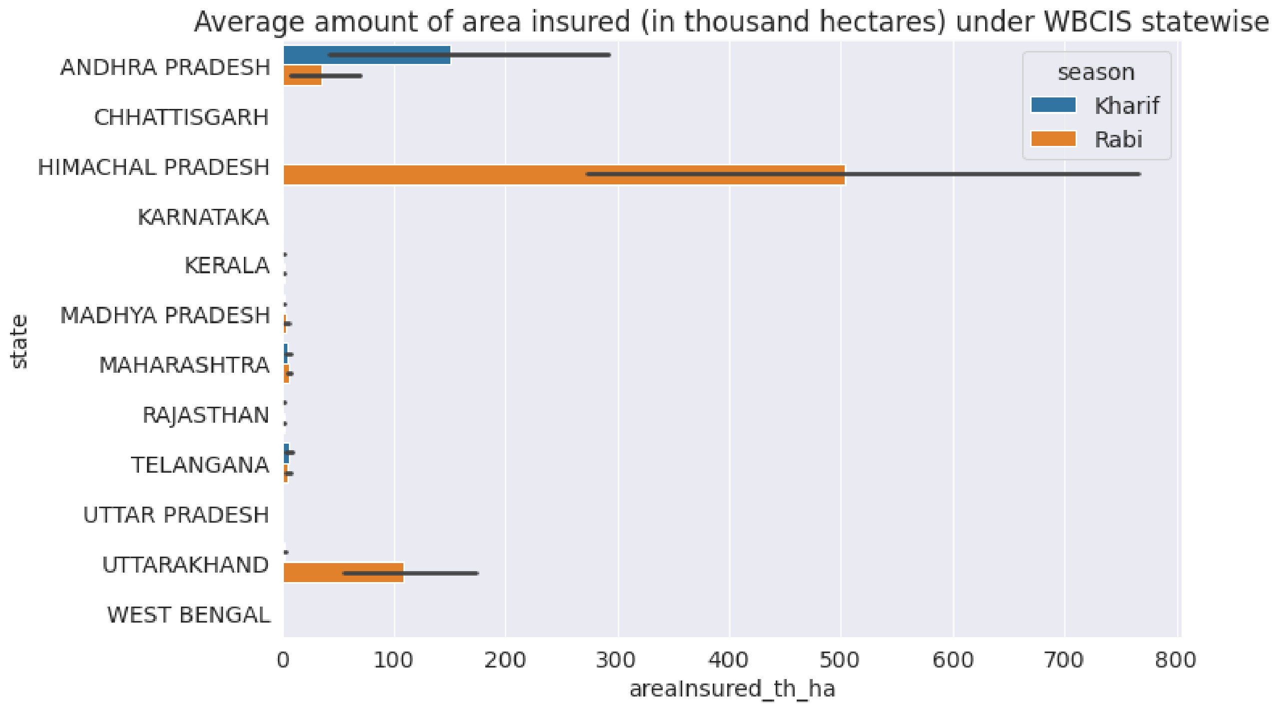
Here we can observe, PMFBY was most preferred in Maharashtra (during the kharif season) and in West Bengal (during the rabi season). Also, Andhra Pradesh is the only state whose average number of registrations for WBCIS (for kharif season) has a huge margin from the other states. Rest of the most of the states prefer PMFBY over WBCIS.

4) Average amount of area insured (in thousand hectares) under each scheme statewise

```
fig, ax = plt.subplots(nrows=2,ncols=1,figsize=(10,15))
sns.barplot(y=pmfby_df.state,x=pmfby_df.areaInsured_th_ha,hue=pmfby_df.season,ax=ax[0])
sns.barplot(y=wbcis_df.state,x=wbcis_df.areaInsured_th_ha,hue=wbcis_df.season,ax=ax[1])
```

```
Text(0.5, 1.0, 'Average amount of area insured (in thousand hectares) under WBCIS statewise')
```



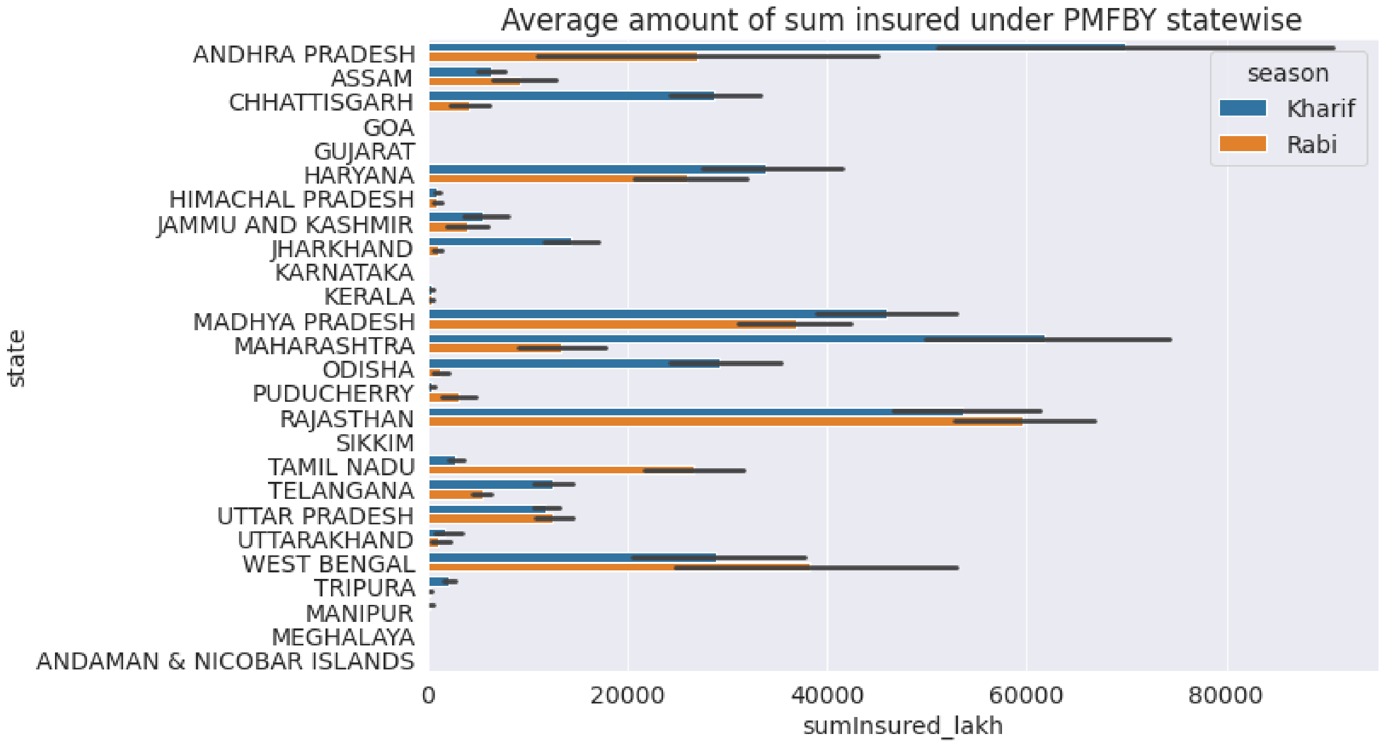


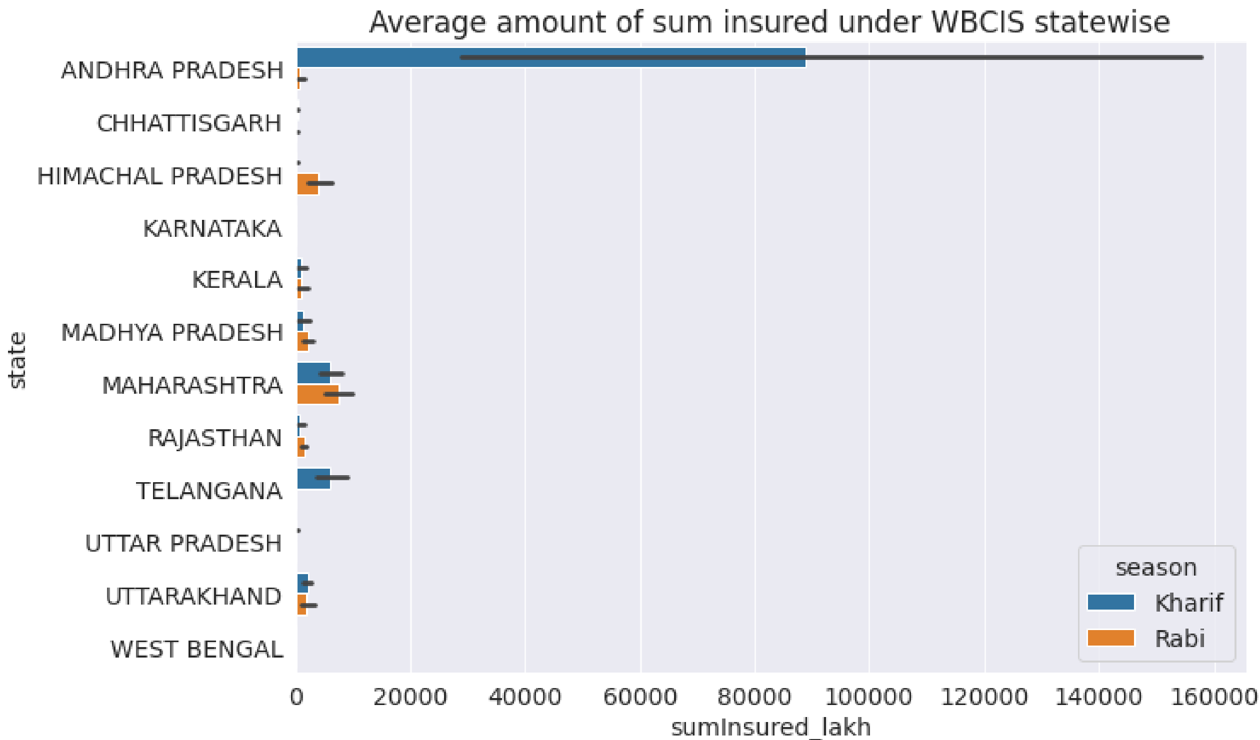
Under PMFBY, Rajasthan has the maximum area insured, followed by Maharashtra and Madhya Pradesh. Also, surprisingly, under WBCIS, as we know that the maximum average registrations were reported in Andhra Pradesh ,but Himachal Pradesh has the maximum amount of area insured. So this implies that majority of the farmers of Himachal Pradesh has a greater amount of land area to be insured. And the majority of the farmers of Andhra Pradesh has comparatively less amount of area to be insured.

5) Average amount of sum insured (in lakhs) under each scheme statewise

```
fig, ax = plt.subplots(nrows=2,ncols=1,figsize=(10,15))
sns.barplot(y=pmfby_df.state,x=pmfby_df.sumInsured_lakh,hue=pmfby_df.season,ax=ax[0]).s
sns.barplot(y=wbcis_df.state,x=wbcis_df.sumInsured_lakh,hue=wbcis_df.season,ax=ax[1]).s
```

```
Text(0.5, 1.0, 'Average amount of sum insured under WBCIS statewise')
```





Under PMFBY, Andhra Pradesh has the maximum amount of sum insured for kharif season but as we can observe in previous plots, it has less number of registrations and less area insured, so this implies that those few farmers have insured their land for a great sum of money. Also, Rajasthan has the maximum amount of sum insured for rabi season. Under WBCIS, again Andhra Pradesh has the maximum amount of sum insured for kharif season and Maharashtra has the maximum amount of sum insured for rabi season.

6) Average premium amount paid by the farmers,state and goi under each scheme

Now, before proceeding ,first we will make dataframes with average value of each state for every feature excluding year.

```
pmfby_mean_df=pmfby_df.drop(columns="year").groupby(["state"]).mean()
pmfby_mean_df
```

state	farmercount	areaInsured_th_ha	sumInsured_lakh	farmerShare_lakh	goiShare_lakh	stateShare_lakh
ANDAMAN & NICOBAR ISLANDS	83.375000	0.067500	46.229687	0.231275	1.602788	2.301737
ANDHRA PRADESH	67838.230769	76.337436	55598.934359	515.479764	1844.693077	2795.373077
ASSAM	14325.829146	11.318291	7726.128230	5.823198	100.660370	198.683809
CHHATTISGARH	28070.875598	45.340670	17032.976729	330.959914	998.062828	998.062828
GOA	45.916667	0.037500	37.764475	0.546125	0.088750	0.088750
GUJARAT	17.653846	0.013846	5.332419	0.142677	0.049223	0.049223
HARYANA	27126.548023	43.670339	30003.789718	656.130508	907.426188	1016.933871
HIMACHAL PRADESH	8660.287500	3.298625	989.464754	17.013316	32.685945	32.685945
JAMMU AND KASHMIR	8515.571429	5.516071	4734.249643	84.837825	168.670357	168.670357
JHARKHAND	20482.947917	13.593646	7692.104062	228.307939	326.057142	459.281204
KARNATAKA	35278.062893	0.035157	0.144687	0.003181	0.013493	0.013733
KERALA	436.378378	0.387838	345.042592	6.754414	8.355823	8.355823
MADHYA PRADESH	46245.111111	116.755217	41421.300435	752.070990	2380.286674	2380.286674
MAHARASHTRA	85315.642857	113.689206	39556.695507	918.989699	3638.846519	3817.435011
MANIPUR	313.066667	0.336000	229.159080	4.610760	10.584413	3.908260
MEGHALAYA	78.100000	0.055000	41.978020	1.653050	0.023670	0.020680
ODISHA	28560.581590	23.828703	15350.385905	303.857971	1138.581584	1138.581584
PUDUCHERRY	2386.307692	2.989231	2021.001308	16.390408	47.424877	63.541800
RAJASTHAN	70146.344697	150.504394	56633.215455	1141.993674	3333.552235	3488.591174
SIKKIM	174.785714	0.030714	30.073921	0.895157	0.069443	0.007714
TAMIL NADU	20455.575758	24.221023	15444.044242	269.116028	1477.068280	2017.470018
TELANGANA	10692.677419	11.517258	9009.702177	175.721694	177.663306	177.663306
TRIPURA	8086.854545	1.557818	1067.485818	2.614478	8.626858	19.440153
UTTAR PRADESH	26841.325000	24.825083	12217.329200	224.832083	377.557025	377.945925
UTTARAKHAND	3980.403846	2.073558	1475.952885	22.515062	11.115453	11.115453
WEST BENGAL	90065.886364	45.933409	33626.482045	198.363264	482.038968	988.125000

```
wbcis_mean_df=wbcis_df.drop(columns="year").groupby(["state"]).mean()
wbcis_mean_df
```

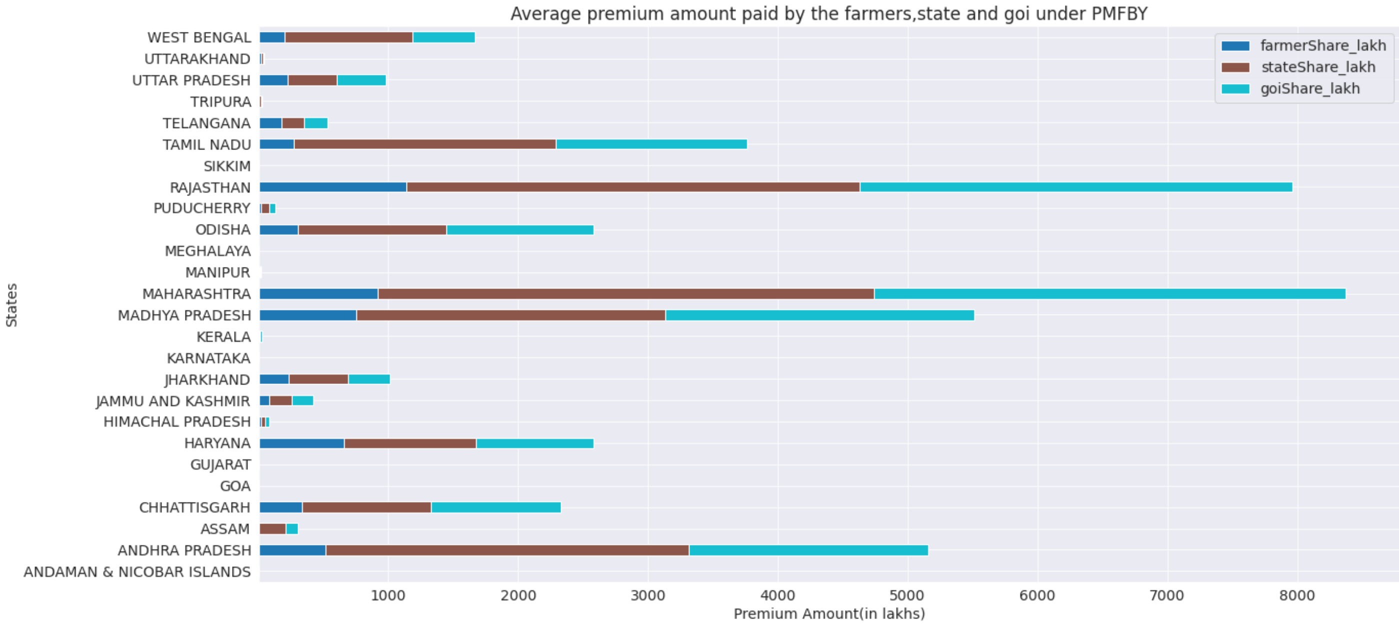
state	farmercount	areaInsured_th_ha	sumInsured_lakh	farmerShare_lakh	goiShare_lakh	stateShare_lakh
ANDHRA PRADESH	52740.903226	102.041613	52027.457097	411.148161	2074.911935	3102.587419
CHHATTISGARH	298.424658	0.277580	276.221142	13.811332	34.296963	57.165279
HIMACHAL PRADESH	4035.493671	280.580506	2353.237109	117.659013	126.227323	143.213525
KARNATAKA	7224.445545	0.002871	0.038930	0.001945	0.004654	0.006429
KERALA	1743.698925	1.354086	1032.326070	19.474348	128.261715	128.499887
MADHYA PRADESH	2703.142105	2.788579	1740.982162	85.616462	52.425831	52.425831
MAHARASHTRA	5688.645455	5.744636	6830.577045	360.511655	710.049498	780.246515
RAJASTHAN	1908.413534	1.327368	1235.696576	61.784843	104.567674	104.567674
TELANGANA	3144.722689	6.143613	3073.759797	153.688078	254.522107	254.522107
UTTAR PRADESH	218.419355	0.121855	125.678286	6.267751	7.034988	7.094720
UTTARAKHAND	2573.989583	60.256875	2064.125260	103.151569	214.162003	214.163175
WEST BENGAL	61.925926	0.015185	24.126385	1.152104	2.085170	2.085170

Now, we will make dataframes having only premium amounts of farmer,state and GOI under each scheme. Then using that dataframes, we will plot stacked bar plot.

```
pmfby_share_df=pmfby_mean_df[["farmerShare_lakh","stateShare_lakh","goiShare_lakh"]]
wbcis_share_df=wbcis_mean_df[["farmerShare_lakh","stateShare_lakh","goiShare_lakh"]]
```

```
pmfby_share_df.plot(kind='barh', stacked=True, colormap='tab10',figsize=(20,10))
plt.title("Average premium amount paid by the farmers,state and goi under PMFBY")
plt.xlabel("Premium Amount(in lakhs)")
plt.ylabel("States")
wbcis_share_df.plot(kind='barh', stacked=True, colormap='tab10',figsize=(20,10))
plt.title("Average premium amount paid by the farmers,state and goi under WBCIS")
plt.xlabel("Premium Amount(in lakhs)")
plt.ylabel("States")
```

```
Text(0, 0.5, 'States')
```



Under PMFBY, Maharashtra has the maximum premium amount but the contribution of state and goi shares are also maximum there. We can observe here that even if the premium amount of Rajasthan is less than the Maharashtra but still the farmer's share in Rajasthan is more than the farmer's share in Maharashtra. Similarly, we can observe this for other states as well. So, the amount of farmer's premium does not only depend on the overall premium amount but on the other factors as well. Under WBCIS, Andhra Pradesh has the maximum premium amount but the contribution of state and goi shares are also maximum there.

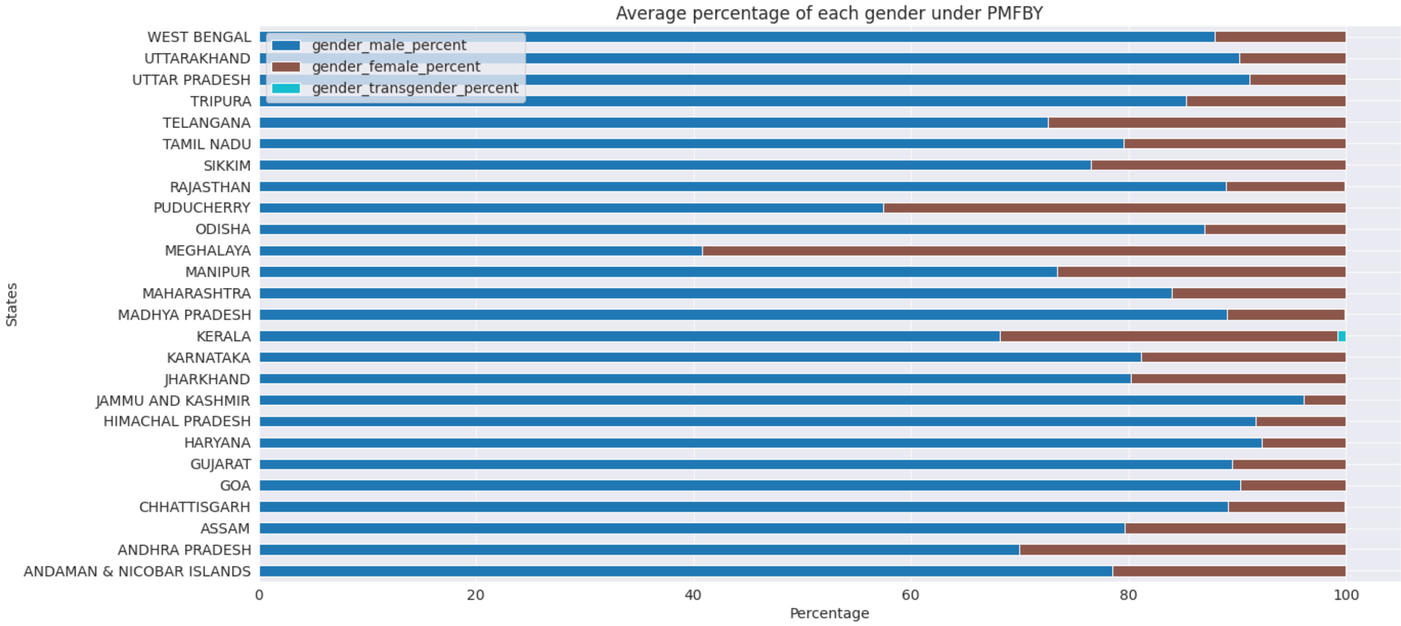
7) Average percentage of each gender under each scheme

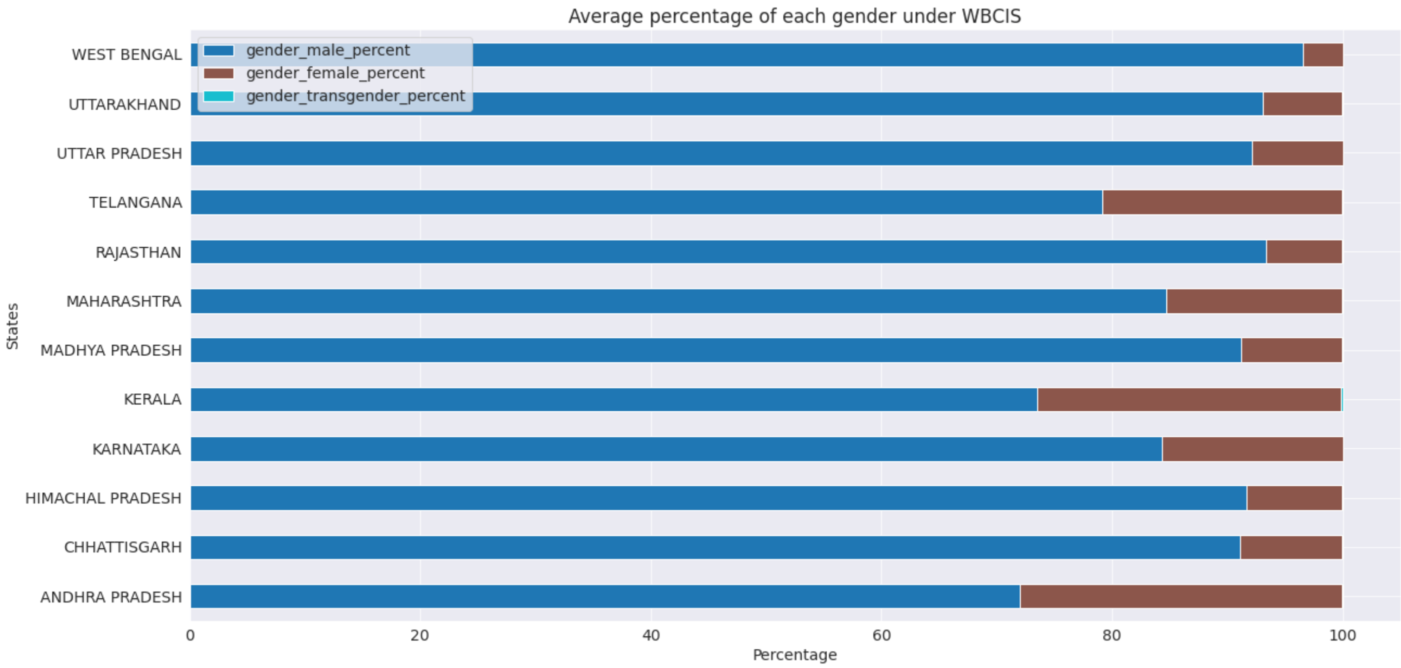
```
pmfby_gender_df=pmfby_mean_df[["gender_male_percent","gender_female_percent","gender_tr
wbcis_gender_df=wbcis_mean_df[["gender_male_percent","gender_female_percent","gender_tr
```

```
pmfby_gender_df.plot(kind='barh', stacked=True, colormap='tab10',figsize=(20,10))
plt.title("Average percentage of each gender under PMFBY")
plt.xlabel("Percentage")
plt.ylabel("States")
wbcis_gender_df.plot(kind='barh', stacked=True, colormap='tab10',figsize=(20,10))
```

```
plt.title("Average percentage of each gender under WBCIS")
plt.xlabel("Percentage")
plt.ylabel("States")
```

```
Text(0, 0.5, 'States')
```





Here, in both the schemes we can clearly observe that the percentage of male farmers is more than the female farmers in every state except Meghalaya. Meghalaya has around 58% of female farmers who has registered for PMFBY. Also, registrations of transgender farmers are negligible for both schemes.

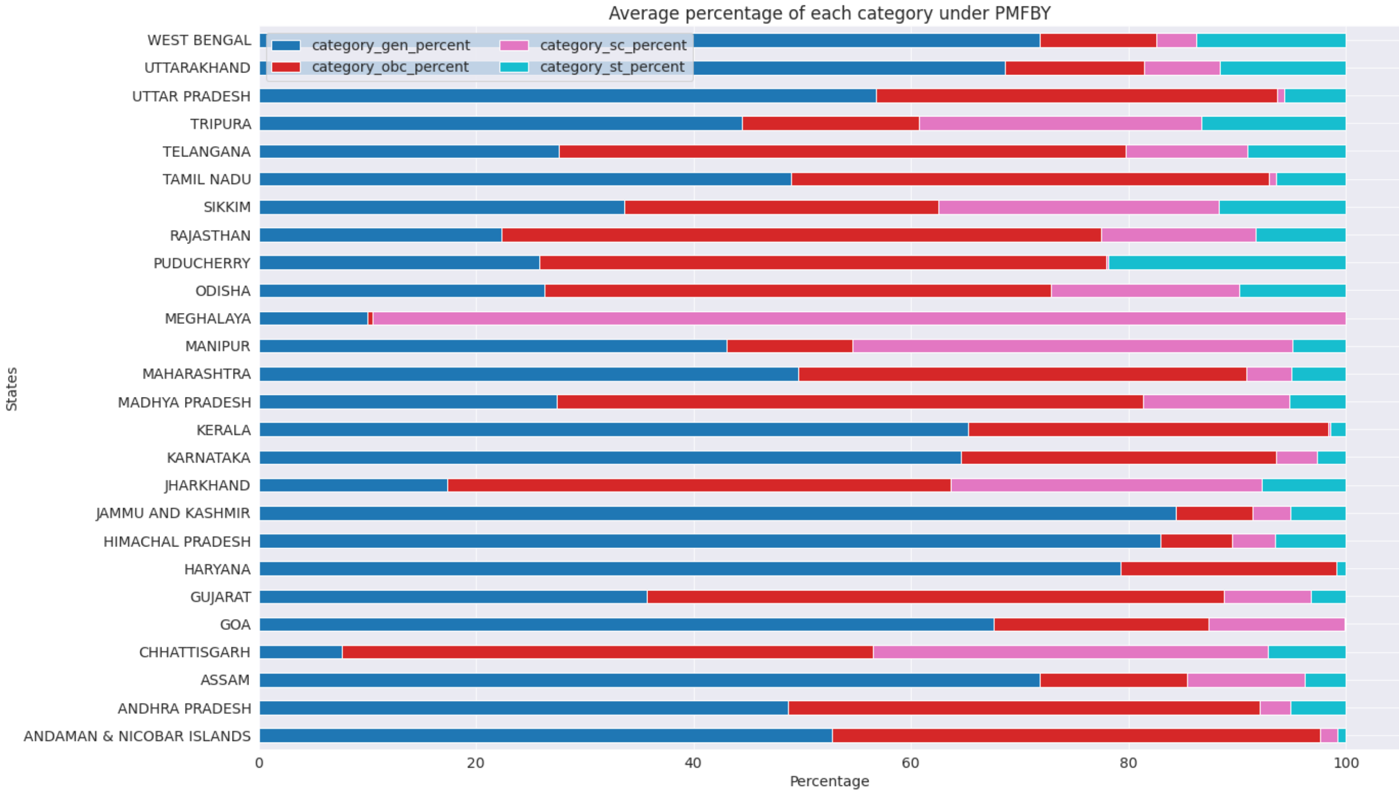
8) Average percentage of different category of farmers under each scheme

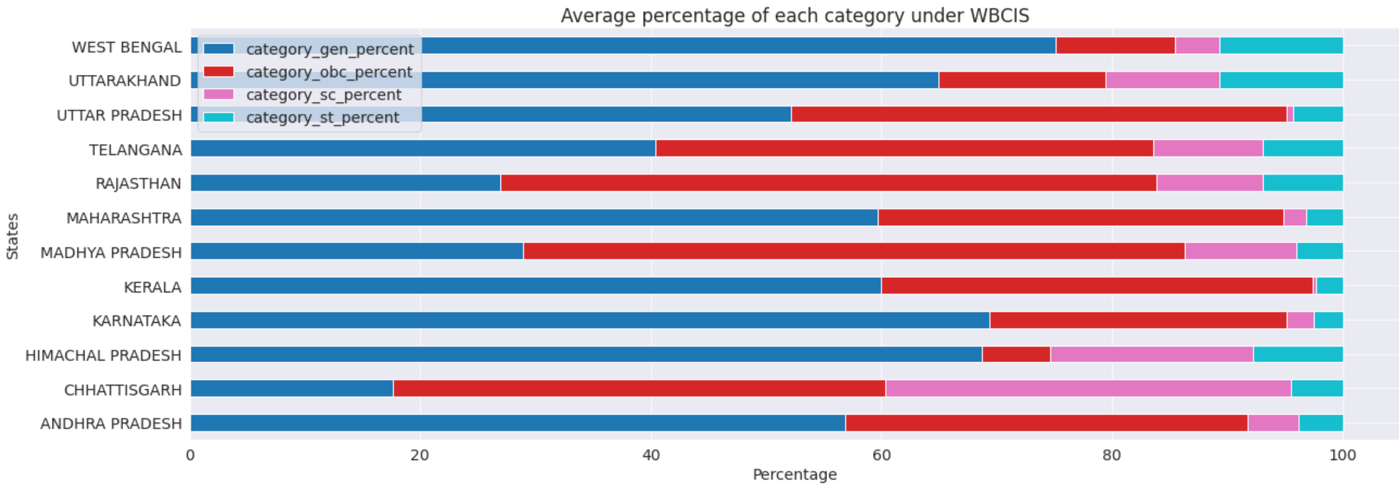
```
pmfby_category_df=pmfby_mean_df[["category_gen_percent","category_obc_percent","categor
wbcis_category_df=wbcis_mean_df[["category_gen_percent","category_obc_percent","categor
```

```
pmfby_category_df.plot(kind='barh', stacked=True, colormap='tab10',figsize=(20,13))
plt.legend(loc="upper left", ncol=2)
plt.title("Average percentage of each category under PMFBY")
```

```
plt.xlabel("Percentage")
plt.ylabel("States")
wbcis_category_df.plot(kind='barh', stacked=True, colormap='tab10',figsize=(20,7))
plt.title("Average percentage of each category under WBCIS")
plt.xlabel("Percentage")
plt.ylabel("States")
```

```
Text(0, 0.5, 'States')
```





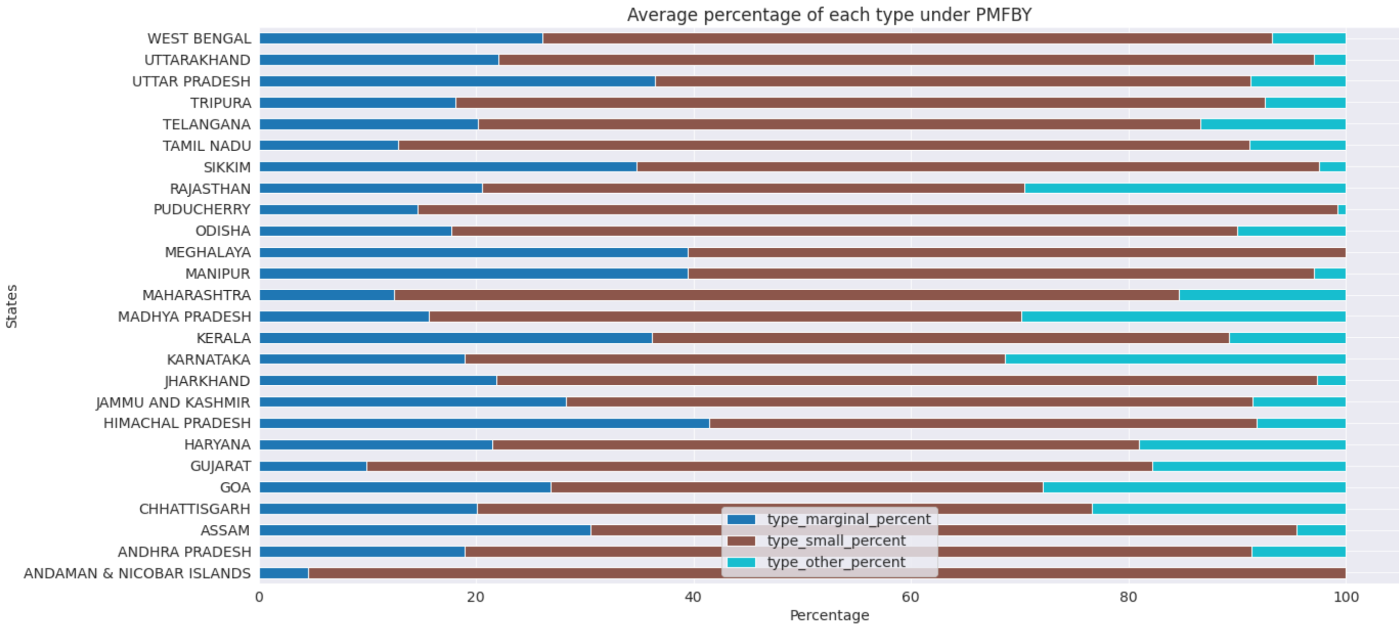
Here we can observe that most of the farmers who registered for both schemes are from the GENERAL category. Then on second position comes the OBC category then comes the SC category and at last the ST category. Though there are some states with exceptions like Meghalaya, around 90% of the farmers in Meghalaya who registered for PMFBY are from SC category.

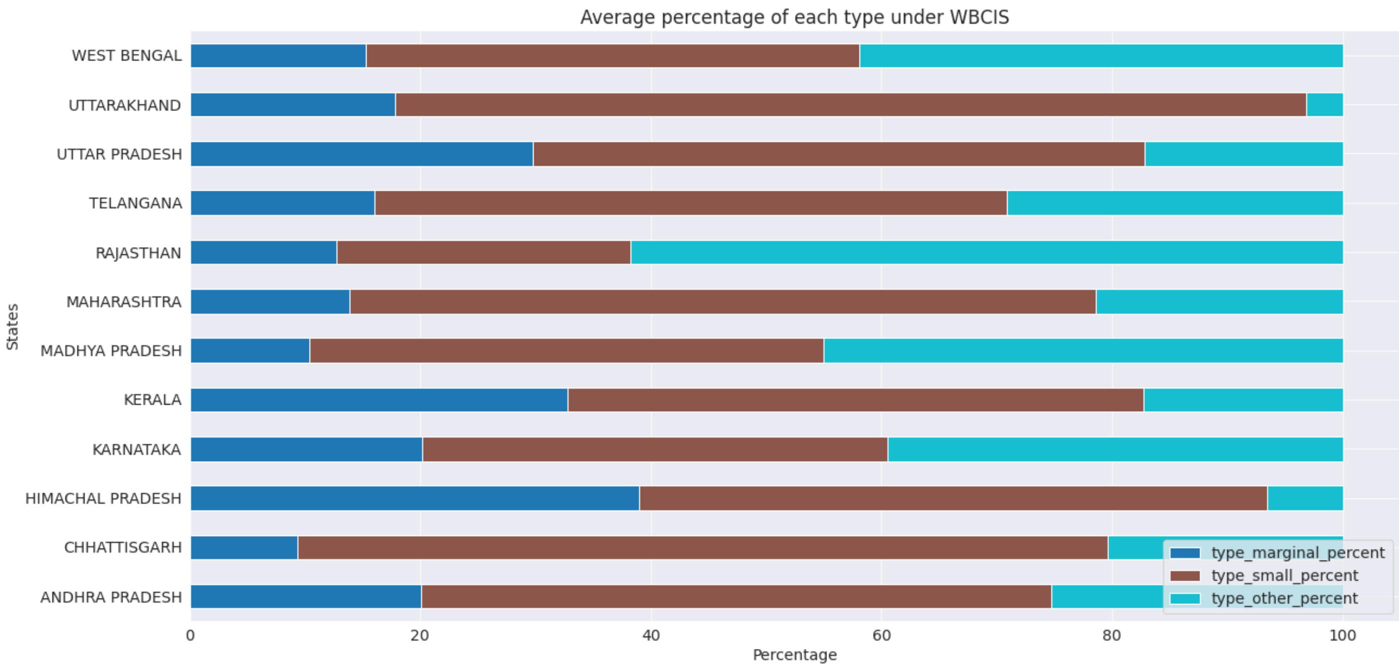
9) Average percentage of different types of farmers under each scheme

```
pmfby_type_df=pmfby_mean_df[["type_marginal_percent","type_small_percent","type_other_p
wbcis_type_df=wbcis_mean_df[["type_marginal_percent","type_small_percent","type_other_p
```

```
pmfby_type_df.plot(kind='barh', stacked=True, colormap='tab10',figsize=(20,10))
plt.title("Average percentage of each type under PMFBY")
plt.xlabel("Percentage")
plt.ylabel("States")
wbcis_type_df.plot(kind='barh', stacked=True, colormap='tab10',figsize=(20,10))
plt.title("Average percentage of each type under WBCIS")
plt.xlabel("Percentage")
plt.ylabel("States")
```

```
Text(0, 0.5, 'States')
```





Under both schemes, most of the farmers are small farmers. But under PMFBY, other type of farmers are least and under WBCIS, marginal type of farmers are least.

Let us save and upload our work to Jovian before continuing

```
import jovian
```

```
jovian.commit()
```

```
[jovian] Updating notebook "saxenaakshit123/exploratory-data-analysis-on-pradhan-mantri-fasal-bima-yojana-pmfby-and-weather-based-crop-d7271" on https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/saxenaakshit123/exploratory-data-analysis-on-pradhan-mantri-fasal-bima-yojana-pmfby-and-weather-based-crop-d7271
```

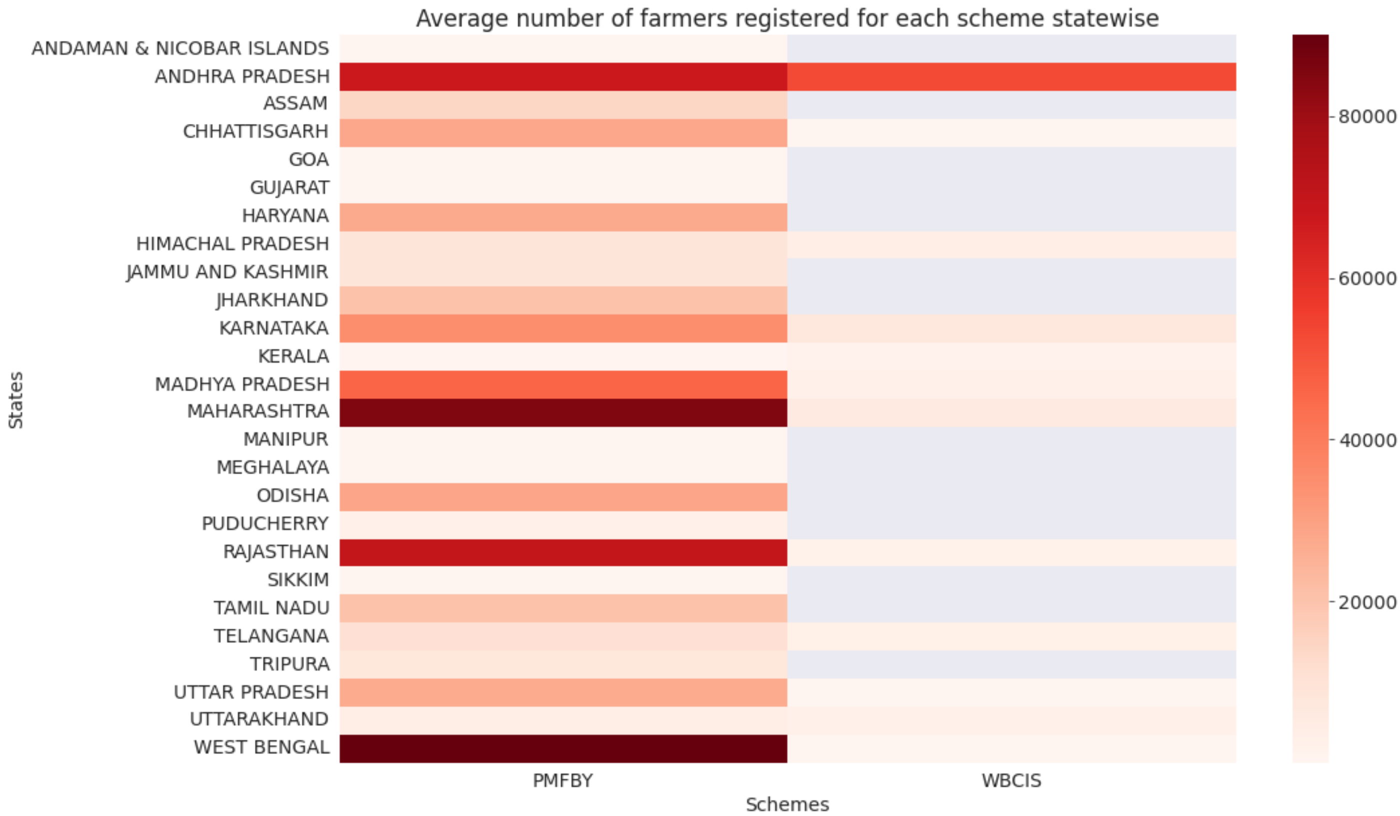
```
'https://jovian.ai/saxenaakshit123/exploratory-data-analysis-on-pradhan-mantri-fasal-bima-yojana-pmfby-and-weather-based-crop-d7271'
```

Asking and Answering Questions

Lets find out some more about these schemes.

Q1: Which scheme is most preferred by the farmers?

```
ndf=df[["scheme","state","farmercount"]].groupby(["scheme","state"]).mean()
ndf.reset_index(inplace=True)
ndf=ndf.pivot(index="state",columns="scheme",values="farmercount")
fig, ax = plt.subplots(figsize=(15, 10))
ax.title.set_text('Average number of farmers registered for each scheme statewise')
fig.patch.set_facecolor('white')
s = sns.heatmap(ndf, cbar=True, cmap='Reds')
s.set(xlabel='Schemes', ylabel='States');
```

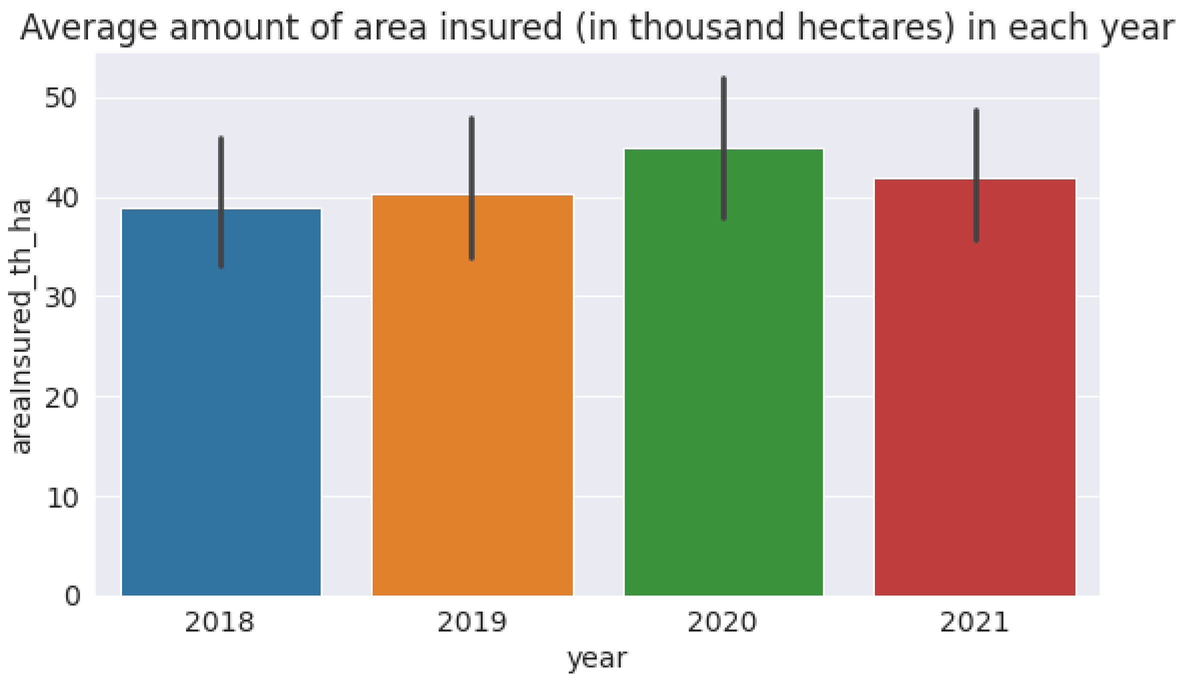


We can clearly observe that the PMFBY is the most preferred scheme as most of the states have either 0 or less than 10000 registrations for WBCIS.

Q2: In which year, maximum amount of area was insured?

```
sns.barplot(x=df.year,y=df.areaInsured_th_ha).set_title("Average amount of area insured
```

```
Text(0.5, 1.0, 'Average amount of area insured (in thousand hectares) in each year')
```



So,in the year 2020, maximum amount of area was insured by the farmers under both schemes together.

Q3: List 5 states with highest gross premium for each scheme.

```
pmfby_gp=pmfby_mean_df["grossPremium_lakh"]
x=pmfby_gp.sort_values(ascending=False)
x.head(5)
```

```
state
MAHARASHTRA       8375.271270
RAJASTHAN         7964.137083
MADHYA PRADESH    5512.644348
ANDHRA PRADESH    5155.545897
TAMIL NADU        3763.654318
Name: grossPremium_lakh, dtype: float64
```

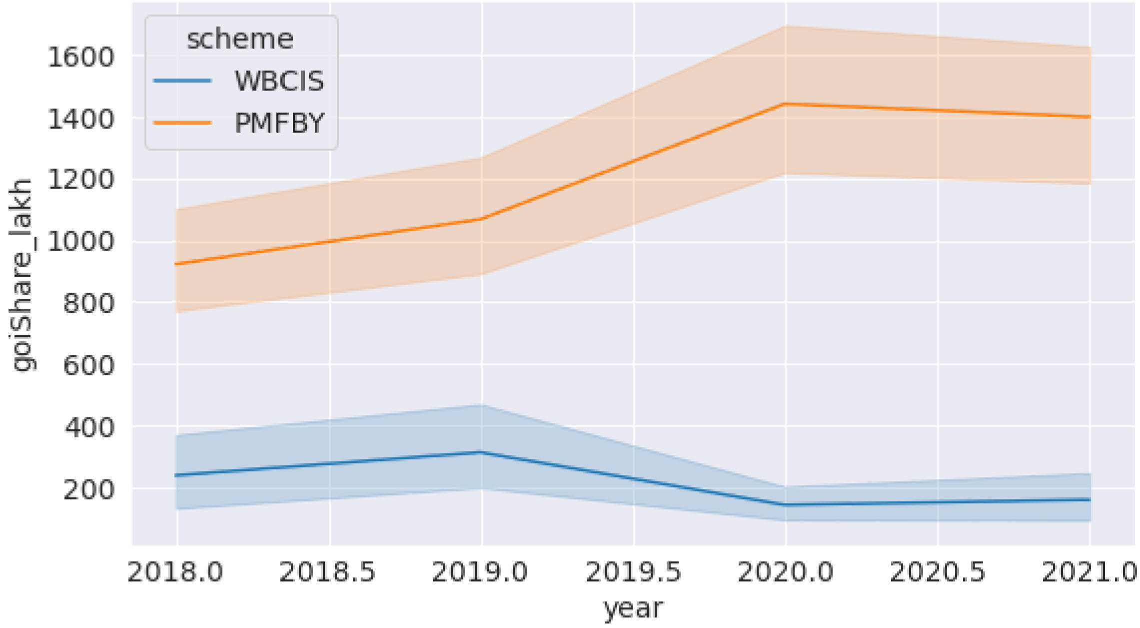
```
wbcis_gp=wbcis_mean_df["grossPremium_lakh"]
y=wbcis_gp.sort_values(ascending=False)
y.head(5)
```

```
state
ANDHRA PRADESH      5588.647419
MAHARASHTRA         1850.807773
TELANGANA            662.732101
UTTARAKHAND          531.476771
HIMACHAL PRADESH     387.100127
Name: grossPremium_lakh, dtype: float64
```

Q4: Show the trend of premium paid by GOI in the past years for each scheme.

```
sns.lineplot(x=df.year,y=df.goiShare_lakh,hue=df.scheme)
```

```
<AxesSubplot:xlabel='year', ylabel='goiShare_lakh'>
```



We can observe that the premium paid by the GOI for WBCIS decreased from the year 2019, and for PMFBY, it showed an increasing trend from the start and got stable from the year 2020. The shaded region is the variation of the share around the trend line.

Q5: List the states having less than 1 thousand hectares of area insured for each scheme.

```
pmfby_area=pmfby_mean_df[["areaInsured_th_ha"]]
pmfby_area[(pmfby_area.areaInsured_th_ha<=1)]
```

	areaInsured_th_ha
state	
ANDAMAN & NICOBAR ISLANDS	0.067500
GOA	0.037500
GUJARAT	0.013846
KARNATAKA	0.035157
KERALA	0.387838
MANIPUR	0.336000
MEGHALAYA	0.055000
SIKKIM	0.030714

```
wbcis_area=wbcis_mean_df[["areaInsured_th_ha"]]
wbcis_area[(wbcis_area.areaInsured_th_ha<=1)]
```

	areaInsured_th_ha
state	
CHHATTISGARH	0.277580

	areaInsured_th_ha
state	
KARNATAKA	0.002871
UTTAR PRADESH	0.121855
WEST BENGAL	0.015185

So,these are the states having less than 1 thousand hectares of area insured. Government should focus to bring the farmlands in these states under insurance.

Let us save and upload our work to Jovian before continuing.

```
import jovian
```

```
jovian.commit()
```

```
[jovian] Updating notebook "saxenaakshit123/exploratory-data-analysis-on-pradhan-mantri-fasal-bima-yojana-pmfby-and-weather-based-crop-d7271" on https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/saxenaakshit123/exploratory-data-analysis-on-pradhan-mantri-fasal-bima-yojana-pmfby-and-weather-based-crop-d7271
```

```
'https://jovian.ai/saxenaakshit123/exploratory-data-analysis-on-pradhan-mantri-fasal-bima-yojana-pmfby-and-weather-based-crop-d7271'
```

Inferences and Conclusion

From the above analysis, we can conclude that

1. Farmers prefer PMFBY scheme more than the WBCIS scheme.
2. Farmers prefer insuring their crops during the Kharif season.
3. Andhra Pradesh is the only state where both scheme is preferred.
4. The percentage of registrations of male farmers is more than the female farmers in every state except Meghalaya.
5. In most of the states, general category has the maximum number of registrations for both the schemes,followed by OBC.
6. Maximum number of registrations are done by the small farmers for both the schemes.

Therefore, there are many states in which farmers hesitate to register for insurance schemes. Government of India and State governments should encourage farmers in those states and should spread awareness among them by educating them the benefits of insuring their crops, so that they don't have to suffer from any loss, when any of the non-preventable natural risks occur in future.

```
import jovian
```

```
jovian.commit()
```

```
[jovian] Updating notebook "saxenaakshit123/exploratory-data-analysis-on-pradhan-
```

```
mantri-fasal-bima-yojana-pmfby-and-weather-based-crop-d7271" on https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/saxenaakshit123/exploratory-data-analysis-on-pradhan-mantri-fasal-bima-yojana-pmfby-and-weather-based-crop-d7271
```

```
'https://jovian.ai/saxenaakshit123/exploratory-data-analysis-on-pradhan-mantri-fasal-bima-yojana-pmfby-and-weather-based-crop-d7271'
```

References and Future Work

More exploration of the dataset can be done using other features present in the dataset as well. The dataset was taken from kaggle - https://www.kaggle.com/datasets/pyatakov/india-pmfby-statistics?select=PMFBY+statistics.csv

```python
import jovian
```

```python
jovian.commit()
```

```
[jovian] Updating notebook "saxenaakshit123/exploratory-data-analysis-on-pradhan-mantri-fasal-bima-yojana-pmfby-and-weather-based-crop-d7271" on https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/saxenaakshit123/exploratory-data-analysis-on-pradhan-mantri-fasal-bima-yojana-pmfby-and-weather-based-crop-d7271
```

```
'https://jovian.ai/saxenaakshit123/exploratory-data-analysis-on-pradhan-mantri-fasal-bima-yojana-pmfby-and-weather-based-crop-d7271'
```