Project Title

Data Engineering Capstone Project

Project Summary

--describe your project at a high level--

The project follows the follow steps:

- Step 1: Scope the Project and Gather Data
- Step 2: Explore and Assess the Data
- Step 3: Define the Data Model
- Step 4: Run ETL to Model the Data
- Step 5: Complete Project Write Up

In [26]:
```
import sql
%load_ext sql
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

In [27]:
```
# Do all imports and installs here
import pandas as pd
import psycopg2
from sql_queries import agmarknet_insert, markets_insert, state_insert, district_insert
import os
```

In [28]:
```
agmarknet_insert
```

Out[28]:
```
'\nINSERT INTO agmarknet (state,district ,market , commodity ,variety, arrival_date  ,min_price    ,max_price  ,modal_price ) VALUES (%s, %s, %s, %s, %s, %s, %s,%s,%s)
'
```

Step 1: Scope the Project and Gather Data

Scope

This projects aims to collect and analyse the AGMARKNET Data, AGMARKNET is one Gov API that publish commoditiy price various mandi in INDIA. There is History data stored Manully htting the API on postman. Aim of the project is to store all the history data into postgres database and update this data base on daly basis using scheduled AWS lambda on daly basis. Nad connct the database to AWS quick sight and anlyse the data basis on daily basis

AGMARNET DATA

Data come from hiiting the GOVT API :

API URL:https://api.data.gov.in/resource/9ef84268-d588-465a-a308-a864a43d0070?api-key=579b464db66ec23bdd000001d9143fc81ac74bce7ad727abc2705a8a&format=csv&limit=10000

In [49]:
```
#Read in the data here
import pandas as pd
df=pd.read_csv('data.csv')
```

In [30]:
```
pd.options.display.max_columns = None
df.head()
```

Out[30]:

	state	district	market	commodity	variety	arrival_date	min_price	max_price	modal_price
0	Andhra Pradesh	Chittor	Mulakalacheruvu	Tomato	Local	06/07/20 0:00	2000.0	3100.0	2100.0
1	Gujarat	Amreli	Damnagar	Bhindi(Ladies Finger)	Bhindi	06/07/20 0:00	900.0	1100.0	1000.0
2	Gujarat	Amreli	Damnagar	Brinjal	Other	06/07/20 0:00	900.0	1100.0	1000.0
3	Gujarat	Amreli	Damnagar	Cabbage	Cabbage	06/07/20 0:00	600.0	800.0	700.0
4	Gujarat	Amreli	Damnagar	Coriander(Leaves)	Coriander	06/07/20 0:00	4900.0	5100.0	5000.0

In [50]:
```
df=df.dropna()
```

In [51]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 903383 entries, 0 to 913928
Data columns (total 11 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   timestamp     903383 non-null  int64
 1   Time          903383 non-null  object
 2   state         903383 non-null  object
 3   district      903383 non-null  object
 4   market        903383 non-null  object
 5   commodity     903383 non-null  object
 6   variety       903383 non-null  object
 7   arrival_date  903383 non-null  object
 8   min_price     903383 non-null  float64
 9   max_price     903383 non-null  float64
 10  modal_price   903383 non-null  float64
dtypes: float64(3), int64(1), object(7)
memory usage: 82.7+ MB
```

Step 2: Explore and Assess the Data

Explore the Data

Identify data quality issues, like missing values, duplicate data, etc.

In [52]:
```
df.describe()
```

Out[52]:

	timestamp	min_price	max_price	modal_price
count	9.033830e+05	903383.000000	903383.000000	903383.000000
mean	1.616207e+09	2948.979808	3448.837857	3220.534537
std	3.840348e+06	3899.387100	5216.771089	4861.902949
min	1.610181e+09	0.000000	0.000000	0.200000
25%	1.612979e+09	1000.000000	1250.000000	1175.000000
50%	1.615970e+09	1900.000000	2250.000000	2050.000000
75%	1.618562e+09	3600.000000	4325.000000	4000.000000
max	1.625075e+09	155000.000000	480000.000000	350000.000000

Daily scrit

In [34]:
```
import requests
import json
import pandas as pd
import urllib3
from sqlalchemy import create_engine
http=urllib3.PoolManager()


url = "https://api.data.gov.in/resource/9ef84268-d588-465a-a308-a864a43d0070?api-key=579b464db66ec23bdd000001d9143fc81ac74bce7ad727abc2705a8a&format=csv&limit=10000"

payload={}
headers = {
  'accept': 'application/xml',
  'Cookie': 'BIGipServerapi.data.gov.in=!3+MbghwZgEPYHt6CbbshOuMRiHS6yPaTO/SoI8x2yuzWQTb260vG8vD/gim86FybvljnTZREBSVVyQ==; TS01a12685=0161d6dfc399201b55df766bb2117226b3
}

response = requests.request("GET", url, headers=headers, data=payload)
x=response.text
listt=x.split("\n")

columns=listt[0].split(",")

State=[]
District=[]
Market=[]
Commodity=[]
Variety=[]
Arrival_Date=[]
Min_x0020_Price=[]
Max_x0020_Price=[]
Modal_x0020_Price=[]

for i in listt[1:-1]:
    State.append(i.split(',')[0])
    District.append(i.split(',')[1])
    Market.append(i.split(',')[2])
    Commodity.append(i.split(',')[3])
    Variety.append(i.split(',')[-5])
    Arrival_Date.append(i.split(',')[-4])
    Min_x0020_Price.append(i.split(',')[-3])
    Max_x0020_Price.append(i.split(',')[-2])
    Modal_x0020_Price.append(i.split(',')[-1])

dff=pd.DataFrame({'State':State,'District':District,'Market':Market,"Commodity":Commodity,"Variety":Variety,"Arrival_Date":Arrival_Date,"Min_Price":Min_x0020_Price,"Max
dff.head(20)
```

Out[34]:

	State	District	Market	Commodity	Variety	Arrival_Date	Min_Price	Max_Price	Modal_Price
0	"Andhra Pradesh"	Chittor	Mulakalacheruvu	Tomato	Local	26/12/2021	4500	5200	5000
1	"Andhra Pradesh"	Kurnool	Allagadda	Jowar(Sorghum)	"Jowar (White)"	26/12/2021	2490	2750	2580
2	"Andhra Pradesh"	Kurnool	Allagadda	Paddy(Dhan)(Common)	Sona	26/12/2021	1860	2180	1950
3	Bihar	"East Champaran"	Chakia	"Bajra(Pearl Millet/Cumbu)"	Bold	26/12/2021	1100	1400	1200
4	Bihar	"East Champaran"	Chakia	Brinjal	"Arkasheela Mattigulla"	26/12/2021	1500	2000	1600
5	Bihar	"East Champaran"	Chakia	Cauliflower	"African Sarson"	26/12/2021	2500	3000	2600
6	Bihar	"East Champaran"	Chakia	Onion	"1st Sort"	26/12/2021	2500	3200	2600
7	Bihar	Gaya	Gaya	Cabbage	Cabbage	26/12/2021	1500	2500	2000
8	Bihar	Gaya	Gaya	Cauliflower	"African Sarson"	26/12/2021	1500	2500	2000
9	Chattisgarh	Dantewada	Gidam	Paddy(Dhan)(Common)	"Paddy Medium"	26/12/2021	1300	1500	1400
10	Chattisgarh	Kanker	Charama	Paddy(Dhan)(Common)	Other	26/12/2021	1500	1500	1500
11	Chattisgarh	Kanker	Lakhanpuri	Paddy(Dhan)(Common)	Other	26/12/2021	1500	1500	1500
12	Chattisgarh	Kanker	Narharpur	Paddy(Dhan)(Common)	Other	26/12/2021	1500	1500	1500
13	Chattisgarh	Mahasamund	Bagbahra	Paddy(Dhan)(Common)	MTU-1001	26/12/2021	1380	1380	1380
14	Chattisgarh	Mahasamund	Bagbahra	Paddy(Dhan)(Common)	"Swarna Masuri (New)"	26/12/2021	1400	1400	1400
15	Chattisgarh	Narayanpur	Narayanpur	Maize	Medium	26/12/2021	1300	1400	1350
16	Chattisgarh	Narayanpur	Narayanpur	Paddy(Dhan)(Common)	"Paddy Coarse"	26/12/2021	1200	1300	1250
17	Chattisgarh	Sukma	Doranpal	Paddy(Dhan)(Common)	"Paddy Coarse"	26/12/2021	1200	1250	1220
18	Chattisgarh	Sukma	Konta	Paddy(Dhan)(Common)	"Paddy Coarse"	26/12/2021	1200	1250	1220
19	Chattisgarh	Sukma	Kukanar	Paddy(Dhan)(Common)	"Paddy Coarse"	26/12/2021	1200	1250	1220

Cleaning Steps

Document steps necessary to clean the data

In [35]:
```
df.head()
```

Out[35]:

	state	district	market	commodity	variety	arrival_date	min_price	max_price	modal_price
0	Andhra Pradesh	Chittor	Mulakalacheruvu	Tomato	Local	06/07/20 0:00	2000.0	3100.0	2100.0
1	Gujarat	Amreli	Damnagar	Bhindi(Ladies Finger)	Bhindi	06/07/20 0:00	900.0	1100.0	1000.0
2	Gujarat	Amreli	Damnagar	Brinjal	Other	06/07/20 0:00	900.0	1100.0	1000.0
3	Gujarat	Amreli	Damnagar	Cabbage	Cabbage	06/07/20 0:00	600.0	800.0	700.0
4	Gujarat	Amreli	Damnagar	Coriander(Leaves)	Coriander	06/07/20 0:00	4900.0	5100.0	5000.0

Step 3: Define the Data Model

3.1 Conceptual Data Model

Map out the conceptual data model and explain why you chose that model

Tables:

table name	columns	description	type
agmarknet	state - district - market - commodity - arrival_date - min_price - max_price - modal_price - stores information related to Mandi Price points	dimension table	
markets	market_id - market_name	stores different market names	fact table
state	state_id - state_name	all state name and ID	fact table
districts	districtid - district_name - state_name	stores all district respective to that state data	fact table

In [36]:
```
#creating district table
```

In [37]:
```
x=df.groupby(['district','state'])[['state']].count()
x.columns=['count']
dis=x.reset_index().drop('count',axis=1)
dis
```

Out[37]:

	district	state
0	Agra	Uttar Pradesh
1	Ahmedabad	Gujarat
2	Ajmer	Rajasthan
3	Alappuzha	Kerala
4	Aligarh	Uttar Pradesh
...	...	...
364	Wayanad	Kerala
365	West District	Tripura
366	West Garo Hills	Meghalaya
367	Yamuna Nagar	Haryana
368	kapurthala	Punjab

369 rows × 2 columns

Step 4: Run Pipelines to Model the Data

4.1 Create the data model

Build the data pipelines to create the data model.

In [38]:
```
# After running create_tables.py, insert the data into the database
conn = psycopg2.connect("host=127.0.0.1 dbname=postgres user=postgres password=postgres")
cur = conn.cursor()
```

In [39]:
```
for index, row in df.head(100).iterrows():
    cur.execute(agmarknet_insert, list(row.values))
    conn.commit()
```

In [40]:
```
for i in df['market'].unique():
    cur.execute(markets_insert, [i])
    conn.commit()
```

In [41]:
```
for i in df['state'].unique():
    cur.execute(state_insert, [i])
    conn.commit()
```

In [42]:
```
for index, row in dis.iterrows():
    cur.execute(district_insert, list(row.values))
    conn.commit()
```

4.2 Data Quality Checks

Explain the data quality checks you'll perform to ensure the pipeline ran as expected. These could include:

- Integrity constraints on the relational database (e.g., unique key, data type, etc.)
- Unit tests for the scripts to ensure they are doing the right thing
- Source/Count checks to ensure completeness

Run Quality Checks

In [43]:
```
# Perform quality checks here
cur.execute("SELECT COUNT(*) FROM agmarknet")
conn.commit()
if cur.rowcount < 1:
    print("No data found in table agmarknet")

cur.execute("SELECT COUNT(*) FROM markets")
conn.commit()
if cur.rowcount < 1:
    print("No data found in table markets")

cur.execute("SELECT COUNT(*) FROM state")
conn.commit()
if cur.rowcount < 1:
    print("No data found in table state")

cur.execute("SELECT COUNT(*) FROM district")
conn.commit()
if cur.rowcount < 1:
    print("No data found in table district")
```

In [62]:
```
import sql
%load_ext sql

DB_ENDPOINT = "127.0.0.1"
DB = 'postgres'
DB_USER = 'postgres'
DB_PASSWORD = 'postgres'
DB_PORT = '5432'

# postgresql://username:password@host:port/database
conn_string = "postgresql://{}:{}@{}:{}/{}" \
                        .format(DB_USER, DB_PASSWORD, DB_ENDPOINT, DB_PORT, DB)

print(conn_string)
%sql $conn_string
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
postgresql://postgres:postgres@127.0.0.1:5432/postgres
```

Data Check

In [45]:
```
%%time
%%sql
SELECT *  FROM agmarknet limit 5
```

```
 * postgresql://postgres:***@127.0.0.1:5432/postgres
5 rows affected.
CPU times: user 2.77 ms, sys: 1.46 ms, total: 4.23 ms
Wall time: 3.3 ms
```

Out[45]:

state	district	market	commodity	variety	arrival_date	min_price	max_price	modal_price
Andhra Pradesh	Chittor	Mulakalacheruvu	Tomato	Local	06/07/20 0:00	2000.0	3100.0	2100.0
Andhra Pradesh	Chittor	Mulakalacheruvu	Tomato	Local	06/07/20 0:00	2000.0	3100.0	2100.0
Gujarat	Amreli	Damnagar	Bhindi(Ladies Finger)	Bhindi	06/07/20 0:00	900.0	1100.0	1000.0
Gujarat	Amreli	Damnagar	Brinjal	Other	06/07/20 0:00	900.0	1100.0	1000.0
Gujarat	Amreli	Damnagar	Cabbage	Cabbage	06/07/20 0:00	600.0	800.0	700.0

In [46]:
```
%%time
%%sql
SELECT * FROM markets limit 5
```

```
 * postgresql://postgres:***@127.0.0.1:5432/postgres
5 rows affected.
CPU times: user 3.77 ms, sys: 2.11 ms, total: 5.88 ms
Wall time: 4.49 ms
```

Out[46]:

market_id	market_name
1	Mulakalacheruvu
2	Damnagar
3	Rajpipla
4	Pehowa
5	Bilaspur

In [47]:
```
%%time
%%sql
SELECT * FROM district limit 5
```

```
 * postgresql://postgres:***@127.0.0.1:5432/postgres
5 rows affected.
CPU times: user 4.89 ms, sys: 2.49 ms, total: 7.38 ms
Wall time: 5.63 ms
```

Out[47]:

districtid	district_name	state_name
1	Agra	Uttar Pradesh
2	Ahmedabad	Gujarat
3	Ajmer	Rajasthan
4	Alappuzha	Kerala
5	Aligarh	Uttar Pradesh

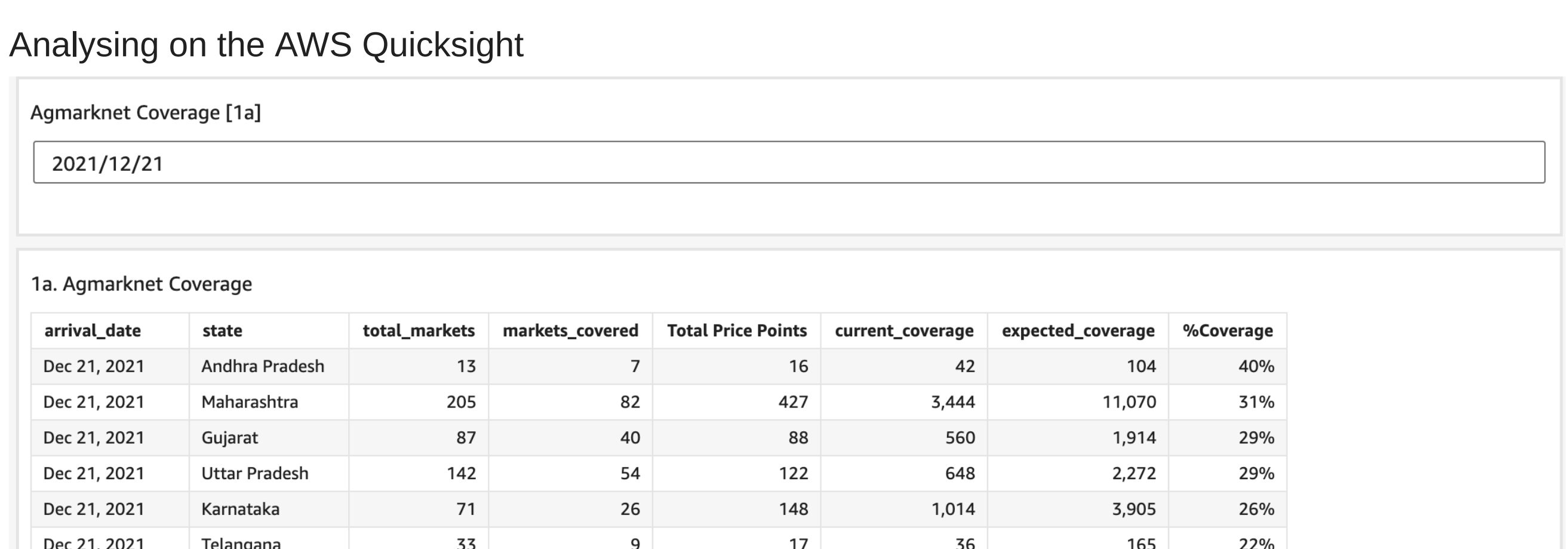
In [48]:
```
%%time
%%sql
SELECT * FROM state limit 5
```

```
 * postgresql://postgres:***@127.0.0.1:5432/postgres
5 rows affected.
CPU times: user 2.88 ms, sys: 1.5 ms, total: 4.38 ms
Wall time: 3.11 ms
```

Out[48]:

stateid	state_name
1	Andhra Pradesh
2	Gujarat
3	Haryana
4	Himachal Pradesh
5	Karnataka

Analysing on the AWS Quicksight

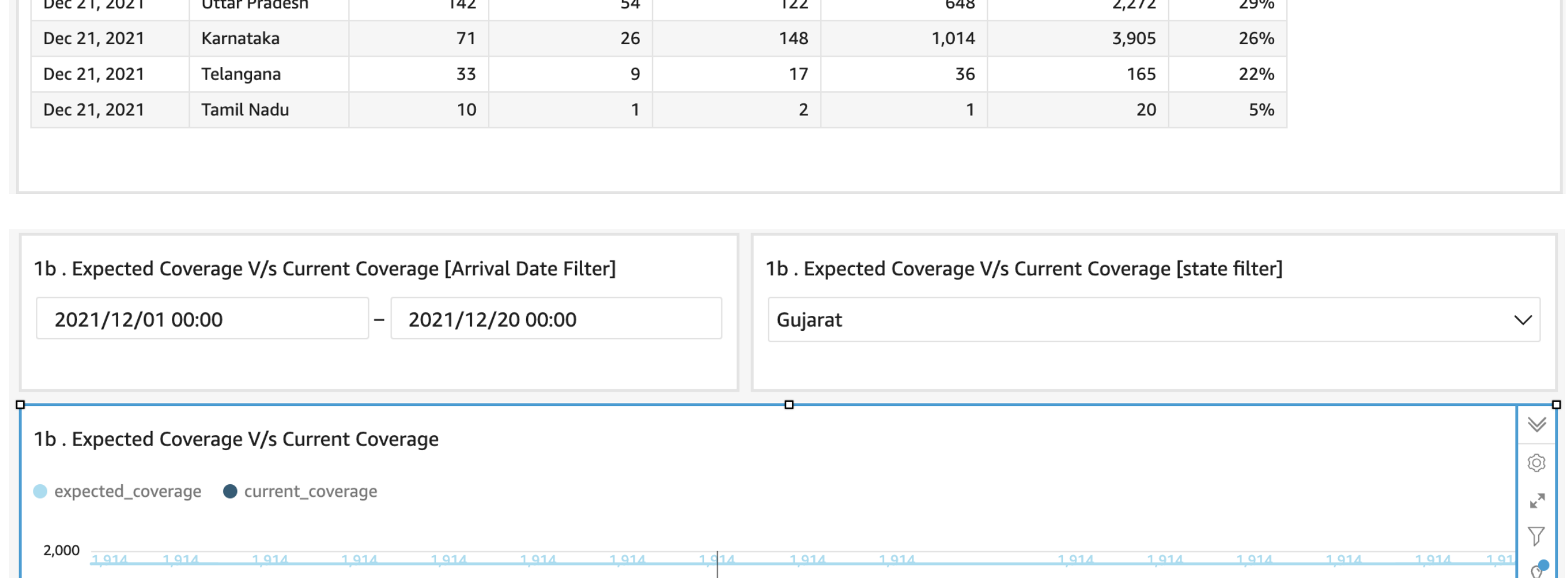


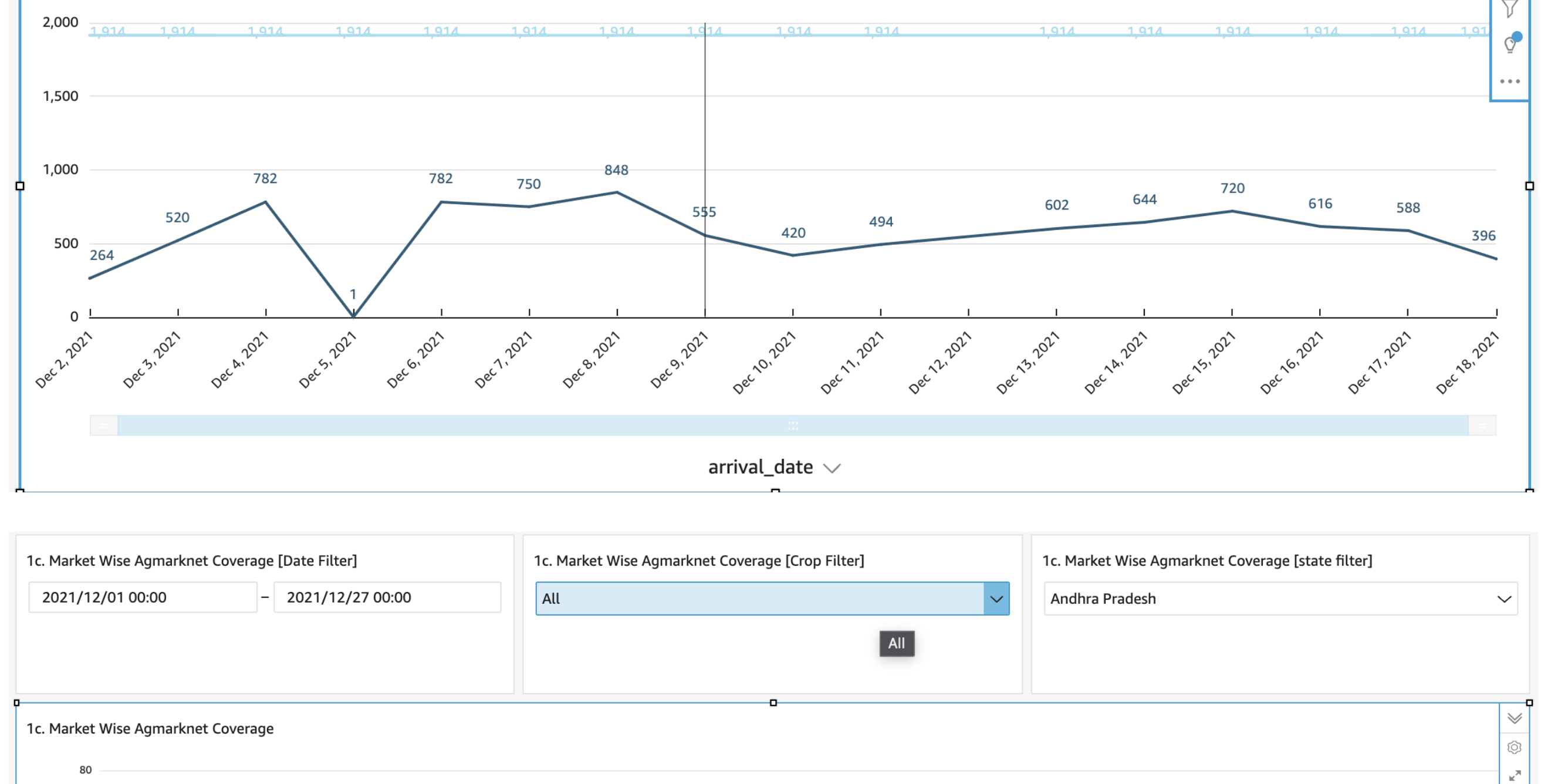
Agmarknet Coverage [1a]

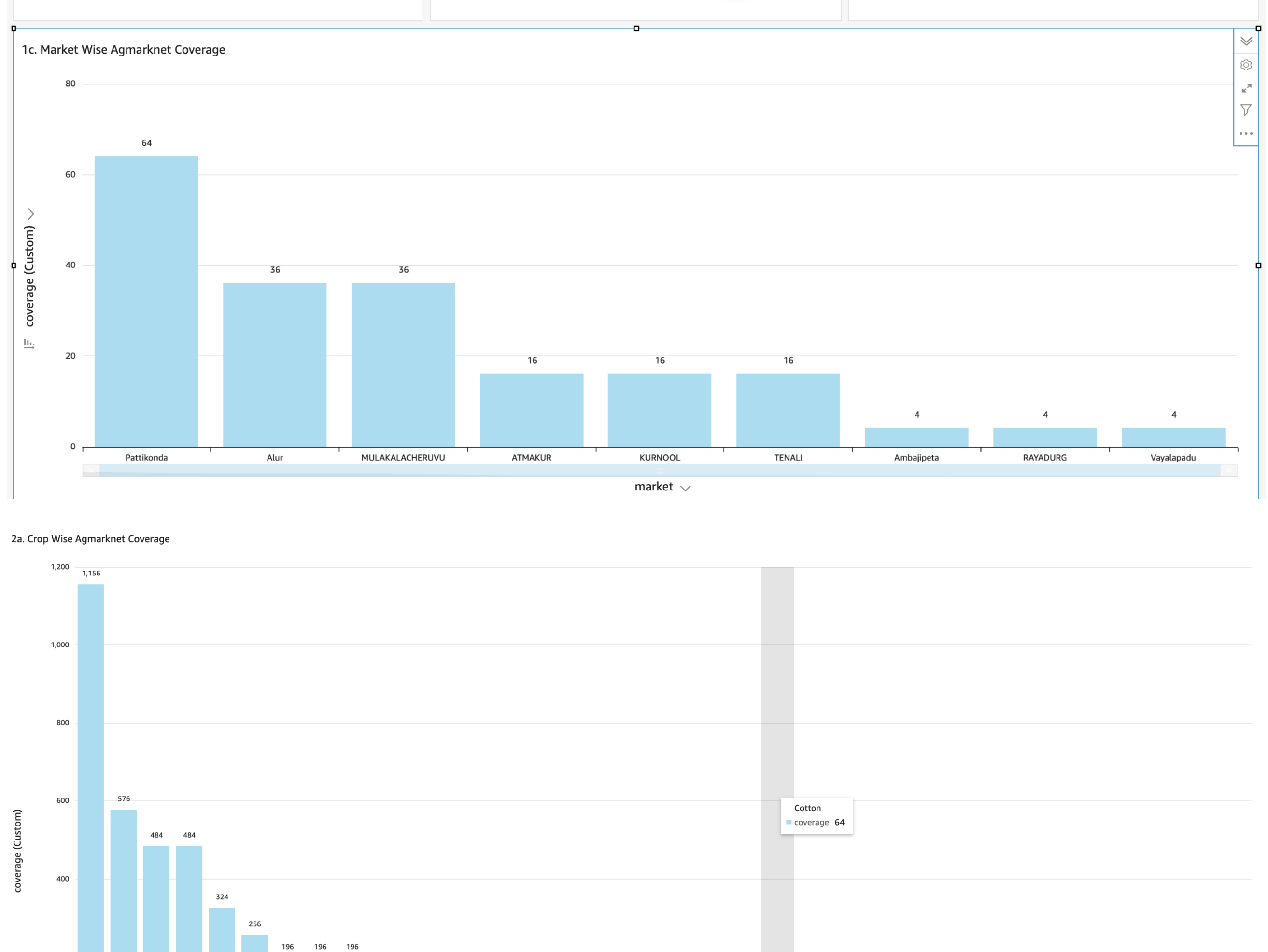
2021/12/21

1a. Agmarknet Coverage

arrival_date	state	total_markets	markets_covered	Total Price Points	current_coverage	expected_coverage	%Coverage
Dec 21, 2021	Andhra Pradesh	13	7	16	42	104	40%
Dec 21, 2021	Maharashtra	205	82	427	3,444	11,070	31%
Dec 21, 2021	Gujarat	87	40	88	560	1,914	29%
Dec 21, 2021	Uttar Pradesh	142	54	122	648	2,272	29%
Dec 21, 2021	Karnataka	71	26	148	1,014	3,905	26%
Dec 21, 2021	Telangana	33	9	17	36	165	22%
Dec 21, 2021	Tamil Nadu	10	1	2	1	20	5%





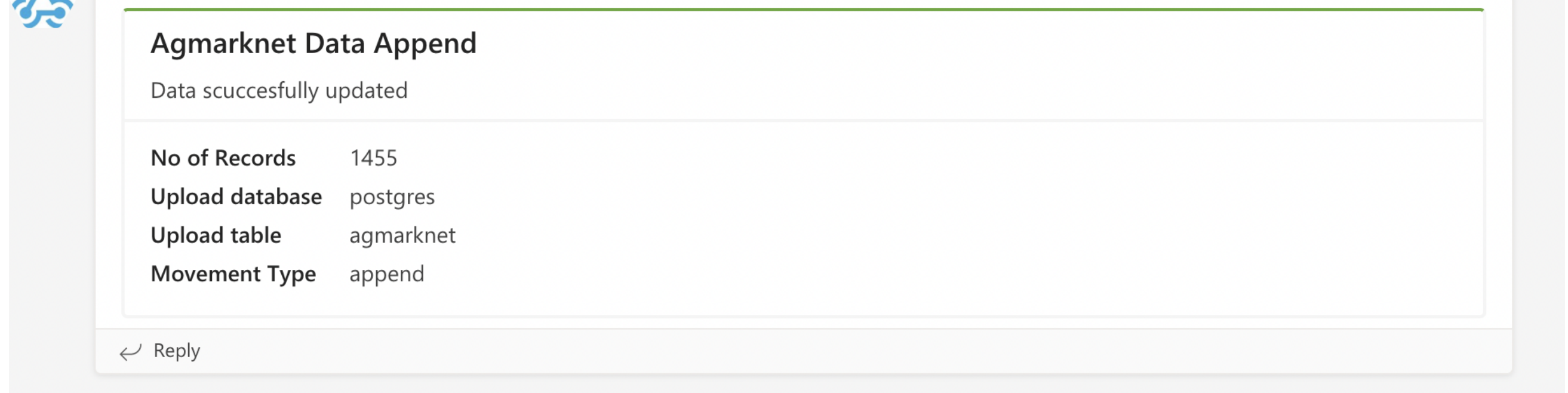


Sheduling and running on daily basis on AWS Lambda Function and Getting Alerts into Microsoft teams

In [64]:
```
import requests
import json
import pandas as pd
import urllib3
from sqlalchemy import create_engine
http=urllib3.PoolManager()
import psycopg2



agmarknet_insert = """
INSERT INTO agmarknet (state,district ,market , commodity ,variety, arrival_date  ,min_price    ,max_price  ,modal_price ) VALUES (%s, %s, %s, %s, %s, %s, %s,%s,%s) ""'

web_hk_url='https://bigdata348.webhook.office.com/webhookb2/67ed8852-6de4-4fa0-ac89-4c9c0bb3a4ce@3e8db0ca-95a6-4914-bac6-6cab03d9aa54/IncomingWebhook/fa2ad6575650437ba
def smsg(nor,db,table,movement_type):
    msg = {
        "@context": "https://schema.org/extensions",
        "@type": "MessageCard",
        "themeColor": "64a837",
        "title": "Agmarknet Data Append",
        "text":f"Data scuccesfully updated ",
        "sections":[
            {
                "facts": [
                    {
                        "name": "No of Records ",
                        "value": nor
                    },
                    {
                        "name": "Upload database",
                        "value": db
                    },
                    {
                        "name": "Upload table",
                        "value": table
                    },
                    {
                        "name":"Movement Type",
                        "value":movement_type
                    }



                ]
            }
        ]
    }
    encoded_msg = json.dumps(msg).encode('utf-8')
    resp = http.request('POST',web_hk_url, body=encoded_msg)


def lambda_handler(event,context):
    url = "https://api.data.gov.in/resource/9ef84268-d588-465a-a308-a864a43d0070?api-key=579b464db66ec23bdd000001d9143fc81ac74bce7ad727abc2705a8a&format=csv&limit=1000

    payload={}
    headers = {
      'accept': 'application/xml',
      'Cookie': 'BIGipServerapi.data.gov.in=!3+MbghwZgEPYHt6CbbshOuMRiHS6yPaTO/SoI8x2yuzWQTb260vG8vD/gim86FybvljnTZREBSVVyQ==; TS01a12685=0161d6dfc399201b55df766bb2117
    }

    response = requests.request("GET", url, headers=headers, data=payload)
    x=response.text
    listt=x.split("\n")

    columns=listt[0].split(",")

    State=[]
    District=[]
    Market=[]
    Commodity=[]
    Variety=[]
    Arrival_Date=[]
    Min_x0020_Price=[]
    Max_x0020_Price=[]
    Modal_x0020_Price=[]

    for i in listt[1:-1]:
        State.append(i.split(',')[0])
        District.append(i.split(',')[1])
        Market.append(i.split(',')[2])
        Commodity.append(i.split(',')[3])
        Variety.append(i.split(',')[-5])
        Arrival_Date.append(i.split(',')[-4])
        Min_x0020_Price.append(i.split(',')[-3])
        Max_x0020_Price.append(i.split(',')[-2])
        Modal_x0020_Price.append(i.split(',')[-1])

    df=pd.DataFrame({'State':State,'District':District,'Market':Market,"Commodity":Commodity,"Variety":Variety,"Arrival_Date":Arrival_Date,"Min_Price":Min_x0020_Price,'

    conn = psycopg2.connect("host=127.0.0.1 dbname=postgres user=postgres password=postgres")
    cur = conn.cursor()
    for index, row in df.iterrows():
        cur.execute(agmarknet_insert, list(row.values))
        conn.commit()

    nor=len(df)
    db='postgres'
    table='agmarknet'
    movement_type='append'
    smsg(nor,db,table,movement_type)
```

In [65]:
```
lambda_handler('a','b')
```

Sample ALERT



In [ ]: