AGRICULTURE : FROM FIELDS TO FUTURE

1. Python code for synthetic data generation :

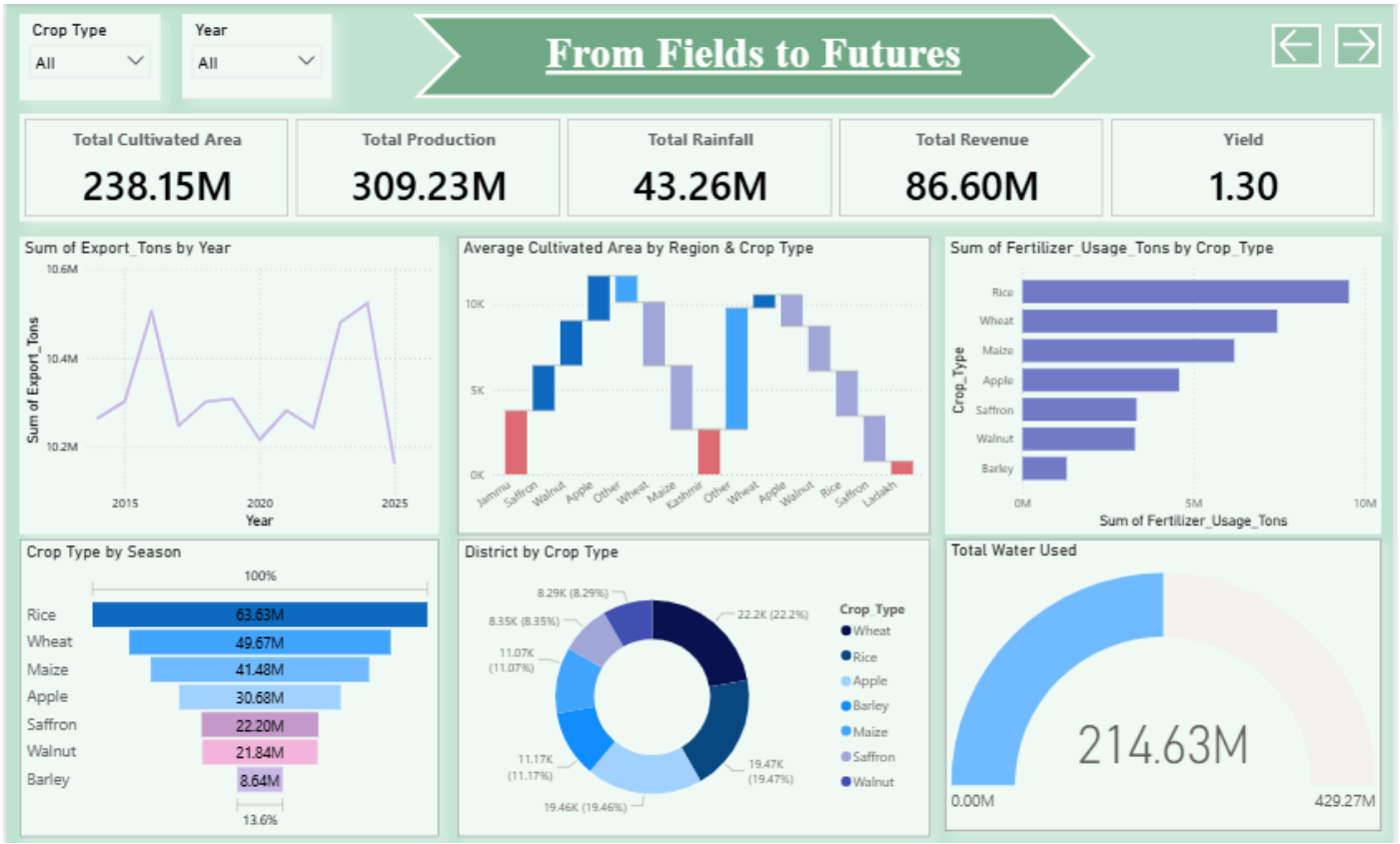
```
import pandas as pd
import numpy as np
import random
from datetime import datetime, timedelta

# Number of rows
num_rows = 100000

# Year range
years = list(range(2014, 2026))
time_period_map = {year: "Past" if year < 2019 else "Present" for year in years}

# Regions & Districts
regions = ["Jammu", "Kashmir", "Ladakh"]
districts = {
    "Jammu": ["Doda", "Jammu", "Kathua", "Kishtwar", "Poonch",
              "Rajouri", "Reasi", "Ramban", "Samba", "Udhampur"],
    "Kashmir": ["Anantnag", "Bandipora", "Baramulla", "Budgam", "Ganderbal",
                "Kulgam", "Kupwara", "Pulwama", "Shopian", "Srinagar"],
    "Ladakh": ["Leh", "Kargil", "Drass", "Zanskar", "Nubra", "Sankoo", "Turtuk"]
}

# Crop preferences by region
region_crops = {
    "Jammu": ["Rice", "Wheat", "Maize"],
    "Kashmir": ["Rice", "Apple", "Saffron", "Walnut"],
    "Ladakh": ["Wheat", "Barley", "Apple"]
}

# Function to determine season & rainfall based on month and region
def get_season_and_rainfall(date_value, region):
    month = date_value.month
    if region == "Ladakh":
        # Less rainfall in Ladakh
        rainfall = round(random.uniform(50, 200), 1)
    else:
        if 6 <= month <= 10:
            rainfall = round(random.uniform(600, 1200), 1)  # Monsoon
        elif 11 <= month or month <= 4:
            rainfall = round(random.uniform(200, 600), 1)   # Winter
        else:
            rainfall = round(random.uniform(50, 250), 1)    # Summer
    season = "Kharif" if 6 <= month <= 10 else "Rabi" if (11 <= month or month <= 4) else
"Zaid"
    return season, rainfall

data = []

for i in range(1, num_rows + 1):
    year = random.choice(years)
```

```
    region = random.choice(regions)
    district = random.choice(districts[region])
    crop = random.choice(region_crops[region])

    # Random date in the selected year
    start_date = datetime(year, 1, 1)
    end_date = datetime(year, 12, 31)
    random_date = start_date + timedelta(days=random.randint(0, (end_date -
start_date).days))

    season, rainfall = get_season_and_rainfall(random_date, region)

    # Bias values by region
    if region == "Jammu":
        cultivated_area = round(np.random.uniform(500, 7000), 2)
    elif region == "Kashmir":
        cultivated_area = round(np.random.uniform(300, 5000), 2)
    else:  # Ladakh
        cultivated_area = round(np.random.uniform(50, 1500), 2)

    production = round(cultivated_area * np.random.uniform(0.6, 2.0), 2)
    export_tons = round(production * np.random.uniform(0.1, 0.7), 2)

    # Revenue: Kashmir's saffron/walnuts more valuable
    if crop in ["Saffron", "Walnut", "Apple"]:
        revenue = round(export_tons * np.random.uniform(0.5, 2.0), 2)
    else:
        revenue = round(export_tons * np.random.uniform(0.1, 0.8), 2)

    water_usage = round(cultivated_area * np.random.uniform(0.3, 1.5), 2)
    fertilizer_usage = round(cultivated_area * np.random.uniform(0.05, 0.25), 2)

    data.append([
        i, year, random_date.date(), time_period_map[year], region, district, crop,
        season, cultivated_area, production, export_tons, revenue,
        rainfall, water_usage, fertilizer_usage
    ])

# Create DataFrame
columns = [
    "Record_ID", "Year", "Date", "Time_Period", "Region", "District", "Crop_Type",
    "Season", "Cultivated_Area_HA", "Production_MT", "Export_Tons", "Revenue_Cr",
    "Rainfall_mm", "Water_Usage_ML", "Fertilizer_Usage_Tons"
]
df = pd.DataFrame(data, columns=columns)

# Save to CSV
df.to_csv("agriculture.csv", index=False)
print("✅ Improved dataset generated: agriculture.csv")
```

DAX Measures used : for kpis only :

1. Total cultivated area –
   Total Cultivated Area = SUM(Agriculture[Cultivated_Area_HA])
2. Total Production –
   Total Production = SUM(Agriculture[Production_MT])
3. Total Rainfall –
   Total Rainfall = SUM(Agriculture[Rainfall_mm])
4. Yield –
   Yield = DIVIDE(SUM(Agriculture[Production_MT]), SUM(Agriculture[Cultivated_Area_HA]), 0)

DASHBOARD 1:



TRAVEL : JOURNEY THAT SHAPES ECONOMICS

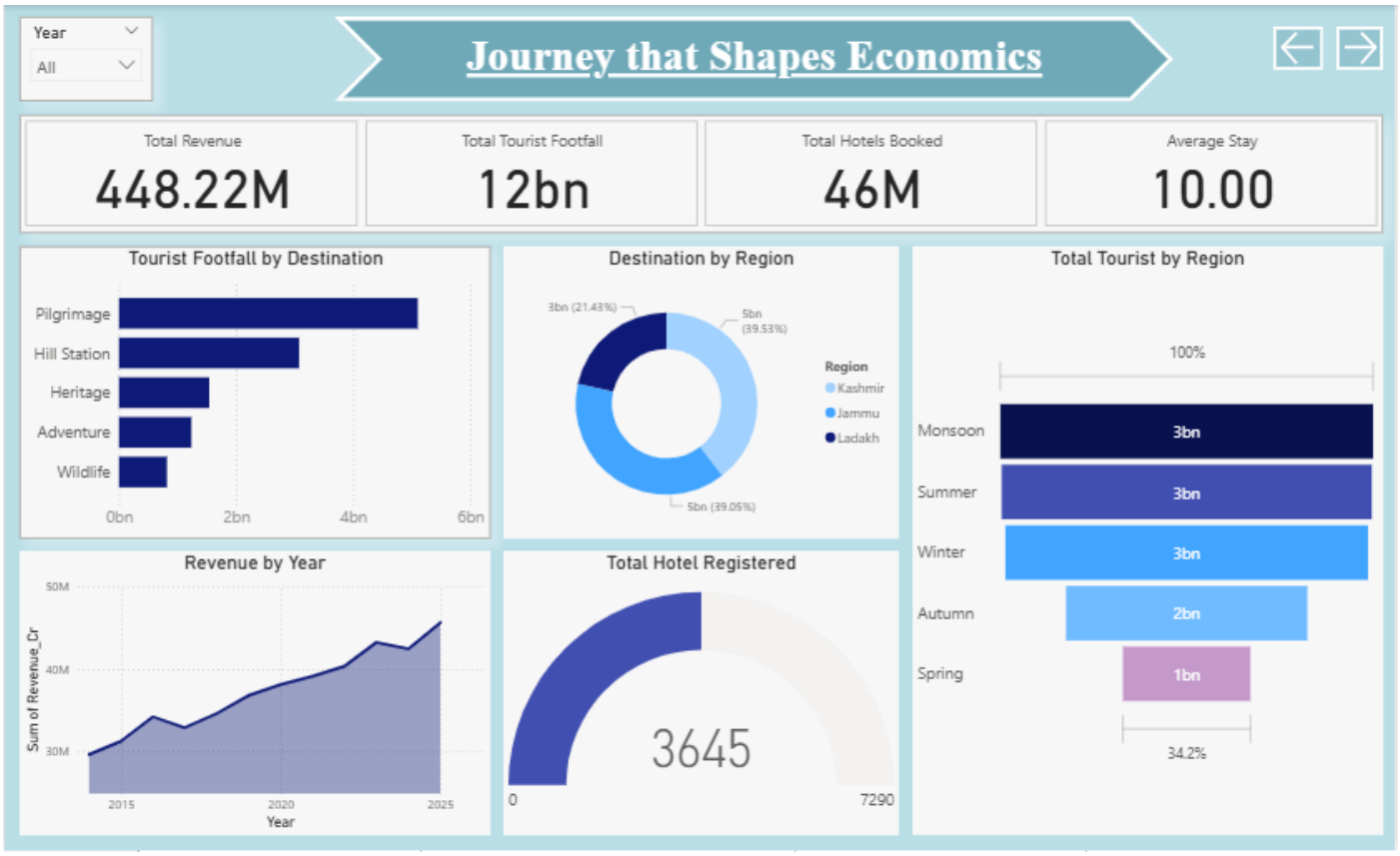
Python code :

```
import random
import pandas as pd
import numpy as np
from datetime import datetime, timedelta

# ----------------------------
# Settings
# ----------------------------
random.seed(42)
np.random.seed(42)

NUM_ROWS = 100000
YEARS = list(range(2014, 2026))  # 2014–2025

# Time period logic
def get_time_period(year):
    return "Past" if year < 2025 else "Present"

# Regions & Districts
regions_districts = {
    "Jammu": ["Doda", "Jammu", "Kathua", "Kishtwar", "Poonch",
              "Rajouri", "Reasi", "Ramban", "Samba", "Udhampur"],
    "Kashmir": ["Anantnag", "Bandipora", "Baramulla", "Budgam", "Ganderbal",
                "Kulgam", "Kupwara", "Pulwama", "Shopian", "Srinagar"],
    "Ladakh": ["Leh", "Kargil", "Drass", "Zanskar", "Nubra", "Sankoo", "Turtuk"]
}

# Tourist types & destination types
tourist_types = ["Domestic", "International"]
destination_types = ["Hill Station", "Heritage", "Wildlife", "Pilgrimage", "Adventure"]

# ----------------------------
# Helper Functions
# ----------------------------
def month_to_season(month):
    if month in (4, 5, 6): return "Summer"
    if month in (7, 8, 9): return "Monsoon"
    if month in (10, 11): return "Autumn"
    if month in (12, 1, 2): return "Winter"
    if month == 3: return "Spring"
    return "Unknown"

def random_date_in_year(year):
    start, end = datetime(year, 1, 1), datetime(year, 12, 31)
    return (start + timedelta(days=random.randint(0, (end - start).days))).date()

def is_festival_season(season, destination_type):
    base_prob = 0.08
    if season in ("Summer", "Winter"): base_prob += 0.12
    if destination_type == "Pilgrimage": base_prob += 0.10
```

```
    return random.random() < min(base_prob, 0.9)

def generate_footfall(region, destination_type, year):
    # Base ranges
    if destination_type == "Pilgrimage": low, high = 5000, 500000
    elif destination_type == "Hill Station": low, high = 2000, 300000
    elif destination_type == "Adventure": low, high = 1000, 100000
    elif destination_type == "Wildlife": low, high = 500, 80000
    else: low, high = 1000, 150000  # Heritage

    # Regional adjustments
    if region == "Ladakh":
        low, high = int(low * 0.3), int(high * 0.5)
        if destination_type == "Adventure":
            low, high = int(low * 2), int(high * 2)  # Ladakh = Adventure hub

    # Yearly growth (4% increase per year since 2014)
    growth_factor = 1 + (year - 2014) * 0.04
    return int(random.randint(low, high) * growth_factor)

def estimate_revenue_crore(footfall, avg_stay_days, tourist_type, festival):
    per_tourist = random.uniform(0.02, 0.08) if tourist_type == "International" else
random.uniform(0.002, 0.02)
    revenue = footfall * per_tourist * (avg_stay_days / 3.0)
    revenue *= random.uniform(0.85, 1.25)
    if festival == "Yes":
        revenue *= 1.2  # festival boost
    return round(max(0.01, revenue), 2)

def estimate_hotels(footfall):
    base = int(max(10, min(2000, footfall // 250)))
    return max(1, int(random.gauss(base, base * 0.25)))

def estimate_employment(hotels_registered, footfall):
    direct = hotels_registered * random.randint(5, 25)
    indirect = int(footfall * random.uniform(0.0005, 0.005))
    return max(10, direct + indirect)

def generate_avg_stay(destination_type, tourist_type):
    base = random.uniform(4, 8) if tourist_type == "International" else random.uniform(1,
6)
    if destination_type == "Adventure": base *= 0.9
    if destination_type == "Pilgrimage": base *= 1.1
    return round(max(1.0, min(10.0, random.gauss(base, 1.2))), 1)

# ----------------------------
# Generate Dataset
# ----------------------------
rows = []
for i in range(1, NUM_ROWS + 1):
    year = random.choice(YEARS)
    date_val = random_date_in_year(year)
    time_period = get_time_period(year)
    region = random.choice(list(regions_districts.keys()))
```

```
    district = random.choice(regions_districts[region])

    # Tourist type (80% Domestic, 20% International; Ladakh gets more foreign visitors)
    if region == "Ladakh":
        tourist_type = np.random.choice(tourist_types, p=[0.6, 0.4])
    else:
        tourist_type = np.random.choice(tourist_types, p=[0.8, 0.2])

    destination_type = random.choice(destination_types)
    season = month_to_season(date_val.month)

    tourist_footfall = generate_footfall(region, destination_type, year)
    avg_stay_days = generate_avg_stay(destination_type, tourist_type)
    festival_season = "Yes" if is_festival_season(season, destination_type) else "No"
    revenue_cr = estimate_revenue_crore(tourist_footfall, avg_stay_days, tourist_type,
festival_season)
    hotels_registered = estimate_hotels(tourist_footfall)
    employment_generated = estimate_employment(hotels_registered, tourist_footfall)

    rows.append({
        "Record_ID": i,
        "Year": year,
        "Date": date_val.isoformat(),
        "Time_Period": time_period,
        "Region": region,
        "District": district,
        "Tourist_Type": tourist_type,
        "Destination_Type": destination_type,
        "Season": season,
        "Tourist_Footfall": tourist_footfall,
        "Average_Stay_Days": avg_stay_days,
        "Festival_Season": festival_season,
        "Revenue_Cr": revenue_cr,
        "Hotels_Registered": hotels_registered,
        "Employment_Generated": employment_generated
    })

# ----------------------------
# Save to CSV
# ----------------------------
df = pd.DataFrame(rows)
df.to_csv("travel.csv", index=False)
print("✅ travel.csv generated with", len(df), "rows and", len(df.columns), "columns.")
```

DAHBOARD 2 :


Year
All
Journey that Shapes Economics
Total Revenue
448.22M
Total Tourist Footfall
12bn
Total Hotels Booked
46M
Average Stay
10.00
Tourist Footfall by Destination
Pilgrimage
Hill Station
Heritage
Adventure
Wildlife
0bn
2bn
4bn
6bn
Destination by Region
3bn (21.43%)
5bn (39.53%)
5bn (39.05%)
Region
Kashmir
Jammu
Ladakh
Total Tourist by Region
100%
Monsoon
3bn
Summer
3bn
Winter
3bn
Autumn
2bn
Spring
1bn
34.2%
Revenue by Year
50M
40M
30M
Sum of Revenue_Cr
2015
2020
2025
Year
Total Hotel Registered
3645
0
7290

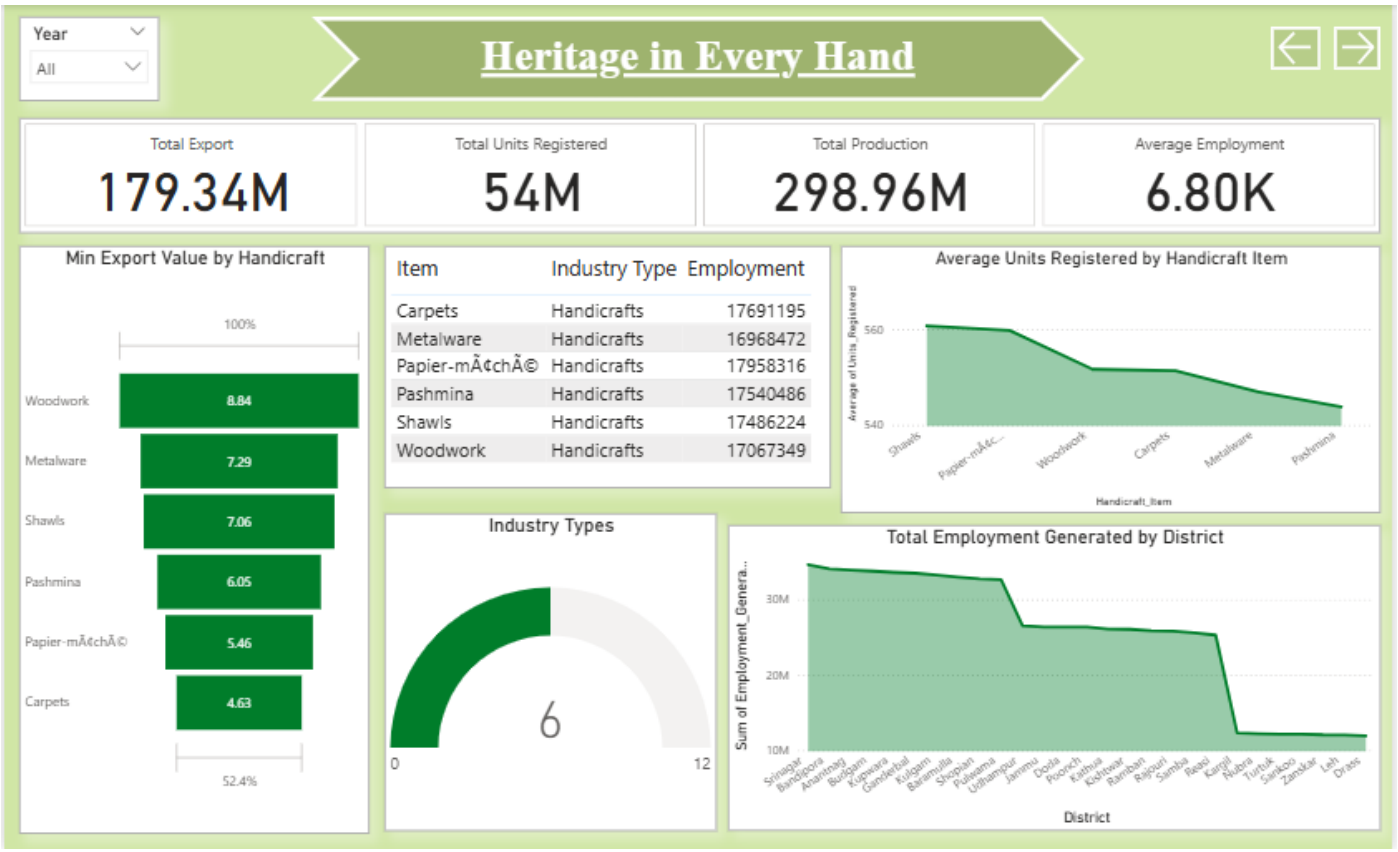
HANDICRAFTS : HERITAGE IN EVERY HANDS
Python code :

```
import pandas as pd
import numpy as np
import random
from datetime import datetime, timedelta

# ----------------------------
# Settings
# ----------------------------
random.seed(42)
np.random.seed(42)

NUM_ROWS = 100000
YEARS = list(range(2014, 2026))  # 2014–2025

# Time period logic: 2025 included in Present
def get_time_period(year):
    if year < 2025:
        return "Past"
    else:
        return "Present"

# Regions & Districts
regions_districts = {
    "Jammu": ["Doda", "Jammu", "Kathua", "Kishtwar", "Poonch",
              "Rajouri", "Reasi", "Ramban", "Samba", "Udhampur"],
    "Kashmir": ["Anantnag", "Bandipora", "Baramulla", "Budgam", "Ganderbal",
                "Kulgam", "Kupwara", "Pulwama", "Shopian", "Srinagar"],
    "Ladakh": ["Leh", "Kargil", "Drass", "Zanskar", "Nubra", "Sankoo", "Turtuk"]
}

industry_types = ["Handicrafts", "Textile", "Food Processing", "Mining", "Tourism
Support", "Small Scale"]
handicraft_items = ["Carpets", "Shawls", "Woodwork", "Papier-mâché", "Metalware",
"Pashmina"]

# Generate random date within a year
def random_date_in_year(year):
    start = datetime(year, 1, 1)
    end = datetime(year, 12, 31)
    delta_days = (end - start).days
    return (start + timedelta(days=random.randint(0, delta_days))).date()

# ----------------------------
# Generate Dataset
# ----------------------------
rows = []
for i in range(1, NUM_ROWS + 1):
    year = random.choice(YEARS)
    date_val = random_date_in_year(year)
    time_period = get_time_period(year)
    region = random.choice(list(regions_districts.keys()))
```

```
    district = random.choice(regions_districts[region])
    industry_type = random.choice(industry_types)

    # Units registered differ by region
    if region == "Jammu":
        units_registered = random.randint(50, 1200)
    elif region == "Kashmir":
        units_registered = random.randint(100, 1500)
    else:  # Ladakh
        units_registered = random.randint(5, 400)

    # Handicrafts more common in Kashmir
    if industry_type == "Handicrafts" and region == "Kashmir":
        handicraft_item = random.choice(handicraft_items)
    elif industry_type == "Handicrafts":
        handicraft_item = random.choice(handicraft_items + ["None"])  # Some non-
handicraft areas
    else:
        handicraft_item = "None"

    # Production & export values scale with units
    production_value = round(units_registered * np.random.uniform(1.0, 10.0), 2)  # crores
    export_value = round(production_value * np.random.uniform(0.3, 0.9), 2)

    # Employment scales with industry size
    employment_generated = int(units_registered * np.random.uniform(5, 20))

    govt_schemes = random.choice(["Yes", "No"])

    rows.append({
        "Record_ID": i,
        "Year": year,
        "Date": date_val.isoformat(),
        "Time_Period": time_period,
        "Region": region,
        "District": district,
        "Industry_Type": industry_type,
        "Handicraft_Item": handicraft_item,
        "Units_Registered": units_registered,
        "Production_Value_Cr": production_value,
        "Export_Value_Cr": export_value,
        "Employment_Generated": employment_generated,
        "Govt_Schemes_Available": govt_schemes
    })

# ----------------------------
# Save to CSV
# ----------------------------
df = pd.DataFrame(rows, columns=[
    "Record_ID", "Year", "Date", "Time_Period", "Region", "District",
    "Industry_Type", "Handicraft_Item", "Units_Registered",
    "Production_Value_Cr", "Export_Value_Cr", "Employment_Generated",
    "Govt_Schemes_Available"
])
```

```
df.to_csv("industry_handicrafts.csv", index=False)
print("✅ industry_handicrafts.csv generated with", len(df), "rows and", len(df.columns),
"columns.")
```

DASHBOARD 3 :



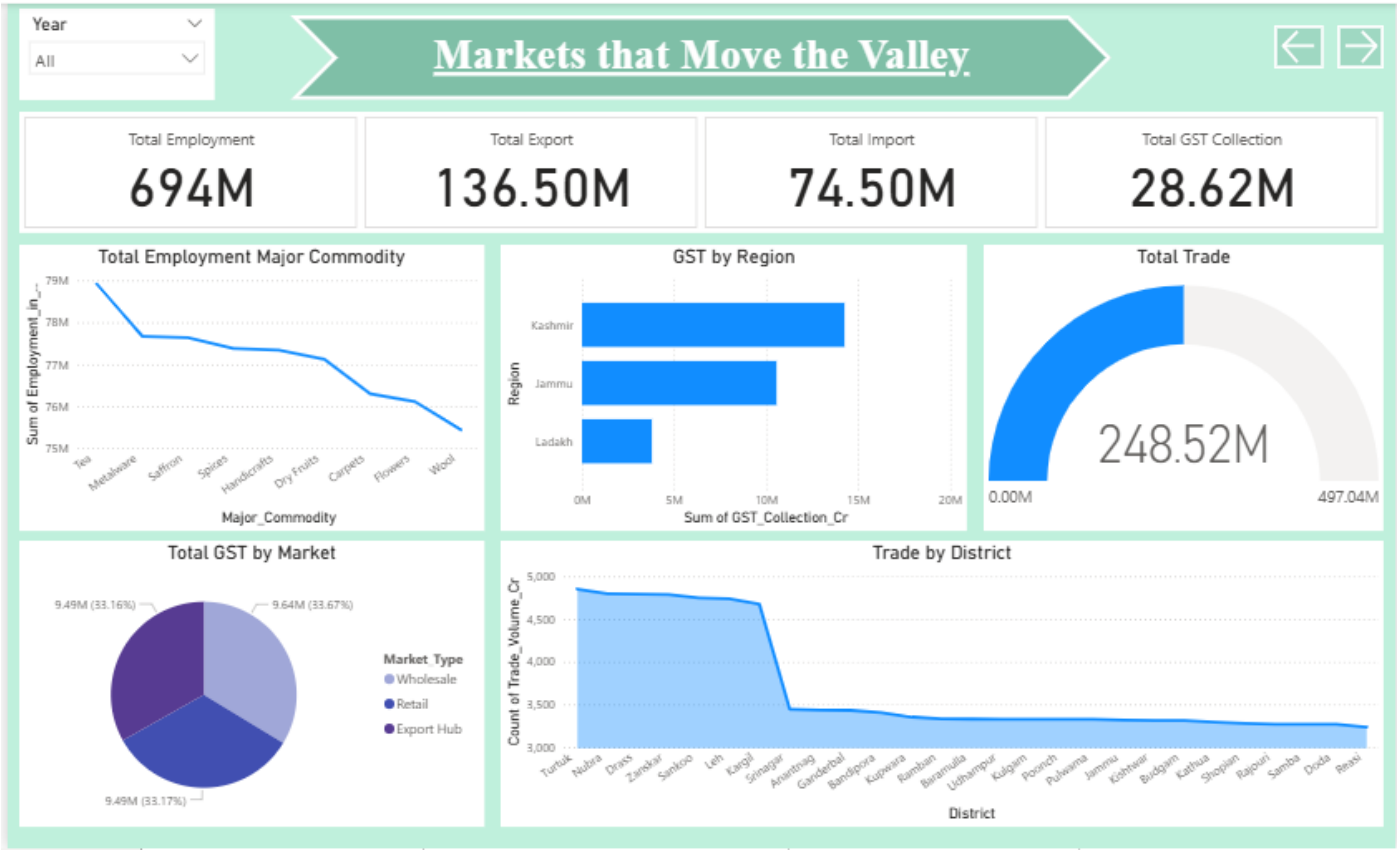
TRADE : MARKETS THAT MOVES VALIES :
Python code :

```
import pandas as pd
import numpy as np
import random
from datetime import datetime, timedelta

# ----------------------------
# Settings
# ----------------------------
random.seed(42)
np.random.seed(42)

NUM_ROWS = 100000
YEARS = list(range(2014, 2026))  # 2014–2025

# Time period logic
def get_time_period(year):
    return "Past" if year < 2025 else "Present"

# Regions & Districts
regions_districts = {
    "Jammu": ["Doda", "Jammu", "Kathua", "Kishtwar", "Poonch",
              "Rajouri", "Reasi", "Ramban", "Samba", "Udhampur"],
    "Kashmir": ["Anantnag", "Bandipora", "Baramulla", "Budgam", "Ganderbal",
                "Kulgam", "Kupwara", "Pulwama", "Shopian", "Srinagar"],
    "Ladakh": ["Leh", "Kargil", "Drass", "Zanskar", "Nubra", "Sankoo", "Turtuk"]
}

# Market types & commodities
market_types = ["Wholesale", "Retail", "Export Hub"]
commodities = ["Saffron", "Handicrafts", "Dry Fruits", "Wool", "Tea", "Spices", "Carpets",
"Metalware", "Flowers"]

# Random date in a given year
def random_date_in_year(year):
    start = datetime(year, 1, 1)
    end = datetime(year, 12, 31)
    return (start + timedelta(days=random.randint(0, (end - start).days))).date()

# ----------------------------
# Generate Dataset
# ----------------------------
rows = []
for i in range(1, NUM_ROWS + 1):
    year = random.choice(YEARS)
    date_val = random_date_in_year(year)
    time_period = get_time_period(year)
    region = random.choice(list(regions_districts.keys()))
    district = random.choice(regions_districts[region])
    market_type = random.choice(market_types)
    commodity = random.choice(commodities)
```

```
    # Region bias in trade volume
    if region == "Jammu":
        base_trade = np.random.uniform(500, 4000)
    elif region == "Kashmir":
        base_trade = np.random.uniform(1000, 5000)
    else:  # Ladakh
        base_trade = np.random.uniform(100, 1500)

    # Commodity boosts (e.g., Saffron & Handicrafts are high value)
    commodity_multiplier = {
        "Saffron": 1.5,
        "Handicrafts": 1.3,
        "Carpets": 1.2,
        "Dry Fruits": 1.1,
        "Wool": 1.0,
        "Tea": 0.9,
        "Spices": 1.0,
        "Metalware": 0.8,
        "Flowers": 0.7
    }[commodity]

    # Year trend → gradual increase in trade volume
    year_factor = 1 + (year - 2014) * 0.03  # ~3% growth per year

    trade_volume = round(base_trade * commodity_multiplier * year_factor, 2)

    export_value = round(trade_volume * np.random.uniform(0.3, 0.8), 2)
    import_value = round(trade_volume * np.random.uniform(0.1, 0.5), 2)

    # Employment scales with market type
    if market_type == "Wholesale":
        employment = random.randint(500, 15000)
    elif market_type == "Export Hub":
        employment = random.randint(1000, 20000)
    else:  # Retail
        employment = random.randint(100, 5000)

    gst_collection = round(trade_volume * np.random.uniform(0.05, 0.18), 2)

    rows.append({
        "Record_ID": i,
        "Year": year,
        "Date": date_val.isoformat(),
        "Time_Period": time_period,
        "Region": region,
        "District": district,
        "Market_Type": market_type,
        "Trade_Volume_Cr": trade_volume,
        "Export_Value_Cr": export_value,
        "Import_Value_Cr": import_value,
        "Major_Commodity": commodity,
        "Employment_in_Trade": employment,
        "GST_Collection_Cr": gst_collection
    })
```

```
# ----------------------------
# Save to CSV
# ----------------------------
df = pd.DataFrame(rows)
df.to_csv("trade_commerce.csv", index=False)
print("✅ trade_commerce.csv generated with", len(df), "rows and", len(df.columns),
"columns.")
```

DASHBOARD 4 :



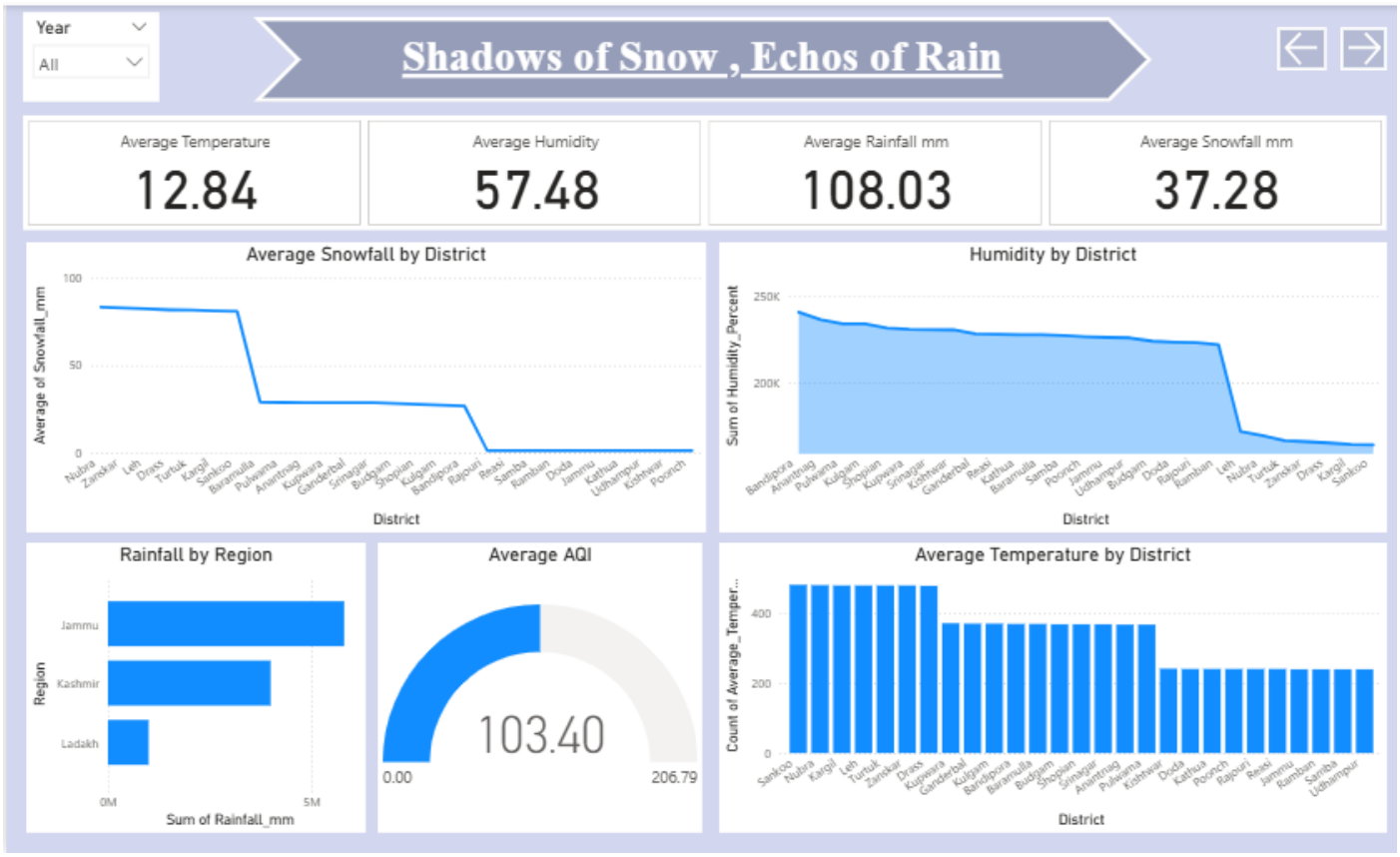
CLIMATE : SHADOWS OF SNOW,ECHOS OF RAIN :
Python code :

```
import random
import pandas as pd
import numpy as np
from datetime import datetime, timedelta

# ----------------------------
# Settings
# ----------------------------
random.seed(42)
np.random.seed(42)

NUM_ROWS = 100000
YEARS = list(range(2014, 2026))  # longer range for climate trends

# Regions & Districts
regions_districts = {
    "Jammu": ["Doda", "Jammu", "Kathua", "Kishtwar", "Poonch",
              "Rajouri", "Reasi", "Ramban", "Samba", "Udhampur"],
    "Kashmir": ["Anantnag", "Bandipora", "Baramulla", "Budgam", "Ganderbal",
                "Kulgam", "Kupwara", "Pulwama", "Shopian", "Srinagar"],
    "Ladakh": ["Leh", "Kargil", "Drass", "Zanskar", "Nubra", "Sankoo", "Turtuk"]
}

# Generate random date within a year
def random_date_in_year(year):
    start = datetime(year, 1, 1)
    end = datetime(year, 12, 31)
    return (start + timedelta(days=random.randint(0, (end - start).days))).date()

# Seasonal mapping
def month_to_season(month):
    if month in (4, 5, 6):
        return "Summer"
    if month in (7, 8, 9):
        return "Monsoon"
    if month in (10, 11):
        return "Autumn"
    if month in (12, 1, 2):
        return "Winter"
    if month == 3:
        return "Spring"
    return "Unknown"

# Generate climate metrics
def generate_temperature(region, year, month):
    warming_trend = (year - 2014) * 0.1  # +0.1°C per year
    if region == "Jammu":
        base = random.uniform(15, 38)
    elif region == "Kashmir":
        base = random.uniform(-6, 30)
    else:  # Ladakh
```

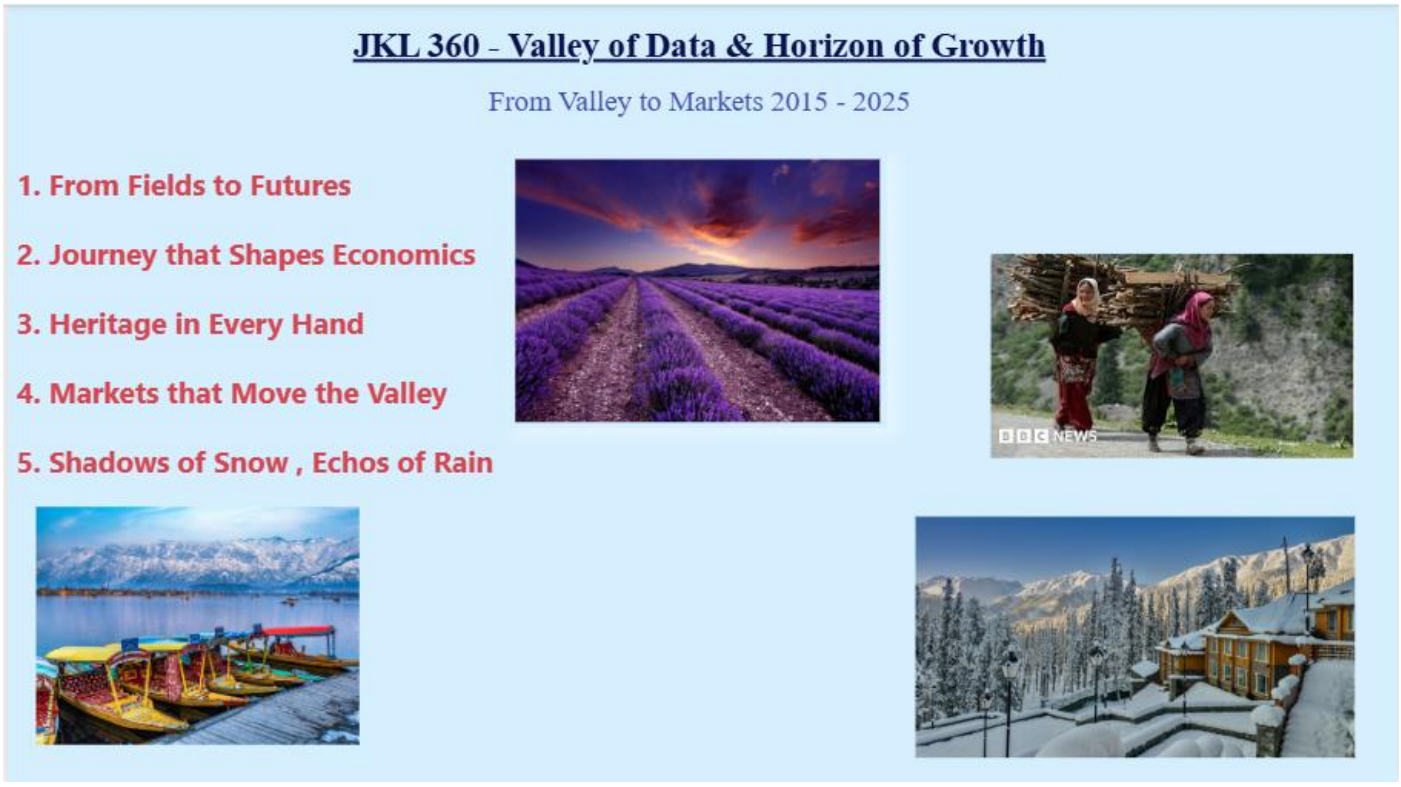
```
        base = random.uniform(-25, 22)
    return round(base + warming_trend, 1)

def generate_rainfall(region, month):
    if region == "Jammu":
        return round(random.uniform(150, 600), 1) if month in [7, 8, 9] else
round(random.uniform(10, 200), 1)
    elif region == "Kashmir":
        return round(random.uniform(80, 400), 1) if month in [7, 8, 9] else
round(random.uniform(10, 150), 1)
    else:  # Ladakh
        return round(random.uniform(0, 60), 1)

def generate_snowfall(region, month):
    if region == "Jammu":
        return round(random.uniform(0, 10), 1) if month in [12, 1, 2] else 0.0
    elif region == "Kashmir":
        return round(random.uniform(20, 150), 1) if month in [12, 1, 2] else
round(random.uniform(0, 20), 1)
    else:  # Ladakh
        return round(random.uniform(50, 300), 1) if month in [11, 12, 1, 2, 3] else
round(random.uniform(0, 30), 1)

def generate_humidity(region):
    if region == "Jammu":
        return round(random.uniform(40, 95), 1)
    elif region == "Kashmir":
        return round(random.uniform(50, 90), 1)
    else:  # Ladakh
        return round(random.uniform(20, 50), 1)

def generate_aqi(region):
    if region == "Jammu":
        return random.randint(70, 250)
    elif region == "Kashmir":
        return random.randint(40, 160)
    else:  # Ladakh
        return random.randint(10, 90)

def extreme_weather(region, season):
    base_prob = 0.05
    if season == "Monsoon" and region == "Jammu":
        base_prob += 0.20  # floods
    if season == "Winter" and region in ["Kashmir", "Ladakh"]:
        base_prob += 0.25  # blizzards
    return "Yes" if random.random() < base_prob else "No"

# ----------------------------
# Generate Dataset
# ----------------------------
rows = []
for i in range(1, NUM_ROWS + 1):
    year = random.choice(YEARS)
    date_val = random_date_in_year(year)
```

```
    month = date_val.month
    season = month_to_season(month)
    region = random.choice(list(regions_districts.keys()))
    district = random.choice(regions_districts[region])

    avg_temp = generate_temperature(region, year, month)
    rainfall = generate_rainfall(region, month)
    snowfall = generate_snowfall(region, month)
    humidity = generate_humidity(region)
    aqi = generate_aqi(region)
    extreme = extreme_weather(region, season)

    rows.append({
        "Record_ID": i,
        "Year": year,
        "Date": date_val.isoformat(),
        "Region": region,
        "District": district,
        "Season": season,
        "Average_Temperature_C": avg_temp,
        "Rainfall_mm": rainfall,
        "Snowfall_mm": snowfall,
        "Humidity_Percent": humidity,
        "Air_Quality_Index": aqi,
        "Extreme_Weather": extreme
    })

# ----------------------------
# Save to CSV
# ----------------------------
df = pd.DataFrame(rows, columns=[
    "Record_ID", "Year", "Date", "Region", "District", "Season",
    "Average_Temperature_C", "Rainfall_mm", "Snowfall_mm",
    "Humidity_Percent", "Air_Quality_Index", "Extreme_Weather"
])

df.to_csv("climate.csv", index=False)
print("✅ Improved climate.csv generated with", len(df), "rows and", len(df.columns),
"columns.")
```

DASHBOARD 5 :



DAHBOARD 6 :



Have all the buttons which will directly take you to the dashboard you want to see