1. A mask was created for all NDCs Associated with Dimethyl Fumarate 240mg and classified as Brand or Generic and named NDC_MASK

```
SELECT  Product_Name, 
        case 
              when Brand_Code = 'Brand' 
              else 'Generic' end as Brand_Generic 
FROM [dbo].[Definitions]
where GPI in ('TEFIDERA')
```

2. The mask was joined to the Medicare 2021 Quarter 3 Pricing table (the latest available when we began this research) to isolate all associated NDCs, Contracts, and reported prices. The data was limited to 30-day supply and named PRICING_FILE.

```
SELECT b.Product_Name, 
b.Brand_Generic, 
a.*
from [Medicare_pricing_file_2021_Q3] a
join NDC_MASK b
on a.NDC=b.NDC
where DAYS_SUPPLY = 30
```

3. The query was joined to the Medicare 2021 Quarter 3 plan information table on contract, plan ID, and Segment ID

```
SELECT a.* 
    ,FORMULARY_ID
from PRICING_FILE a
join [Medicare_plan_information_2021_Q3] b
on a.CONTRACT_ID=b.CONTRACT_ID and a.PLAN_ID=b.PLAN_ID and a_SEGMENT_ID=b_SEGMENT_ID
```

4. The table was then joined to the basic drug formulary table by contract ID, plan ID, segment ID and the tier values added. The table was named TIER

```
SELECT a.* 
    ,TIER_LEVEL_VALUE
from PLAN_INFORMATION a
JOIN [Medicare_basic_drugs_formulary_2021Q3] b
on a.FORMULARY_ID = b.FORMULARY_ID and a.NDC=b.NDC
```

5. The TIER table was aggregated by Contract_Id to get average price and tier level per contract and named BRAND GENERIC

Proprietary database terms have been anonymized
Proprietary database terms have been anonymized.
when a.Brand_Generic is null then 'Generic'
when b.Brand_Generic is null then 'Brand'
else 'Both' end Brand_Generic

FROM BRAND a
FULL JOIN GENERIC b
on a.CONTRACT_ID = b.CONTRACT_ID

8. The Merged table was joined to August Part D enrollment data (the latest available when we added this query to our research) to identify the number of beneficiaries serviced by each contract and named the Lives Table.

Select a.*,
     Plan_Type,
     Organization_Marketing_Name,
     Parent_Organization,
     PartD Lives,
     sum(convert(int,PartD)) OVER(PARTITION BY Parent_Organization) as 'Parent_Total_Org_Lives'
from MERGED a
join [Medicare_Enrollment_Plan_Aug_2021] b
on a.CONTRACT_ID = b.Contrac_Number
where PartD <> '*'

9. A case statement was created from the Lives table to group the Parent Organizations by size

Select *
,case
    When Parent_Total_Org_Lives <= 100000 then 'Small'
    When Parent_Total_Org_Lives <= 1000000 and Parent_Total_Org_Lives > 100000 then 'Medium'
    Else 'Large' end 'size'
from Lives
order by Parent_Organization

10. The table was saved as 'TF_Q3_2021_ANALYSIS.csv'

11. A DataFrame was created from the TF_Q3_2021_ANALYSIS.csv

#import files
file = pd.read_csv('TF_Q3_2021_ANALYSIS.csv')
#create DataFrame
df = pd.DataFrame(file)
12. A column was created to determine lowest price

df['lowest_price_Q3'] = np.where(df['GEN_UNIT_COST'] > 1 , df['GEN_UNIT_COST'], df['BRAND_UNIT_COST'])

13. Determine the percent and count of contract that mandate brand/generic/choice (fig 5)

#group by Brand_Generic Category and sum count
brand_generic = df.groupby('Brand_Generic')['Lives'].sum()
#reindex
brand_generic = brand_generic[['Brand', 'Both', 'Generic']].reset_index()
#create percent column
brand_generic['percent'] = round(brand_generic.Lives/brand_generic.Lives.sum()*100,1)

14. Determine number of Lives by brand/generic/choice (fig 6)

lives_by_mandate = df.groupby(['size', 'Brand_Generic'])[['Lives']].sum()
lives_by_mandate = lives_by_mandate.reset_index()
lives_by_mandate = lives_by_mandate.sort_values(by='Brand_Generic', ascending=False)

15. Separate by organizational size

small = (lives_by_mandate.loc[lives_by_mandate['size'] == 'Small']['Lives'])
small = list(small['Lives'])
medium = (lives_by_mandate.loc[lives_by_mandate['size'] == 'Medium']['Lives'])
medium = list(medium['Lives'])
large = (lives_by_mandate.loc[lives_by_mandate['size'] == 'Large']['Lives'])
large = list(large['Lives'])

16. Create DataFrame for Brand Tier data and charting (fig 6)

brand_tier = df.groupby(['size', 'Brand_Tier'])[['Lives']].sum().reset_index()
17. Create DataFrame for Generic Tier Data and charting (fig 10)

gen_tier = df.groupby(['size','Gen_Tier'])[['Lives']].sum().reset_index()

18. A DataFrame was created to group organizations by size and lives for charting (fig 7)

size_org = org_size.groupby('size').agg({'Parent_Organization':'count','Lives':'sum'})
size_org = size_org.reset_index()

19. A DataFrame was created for a violin plot and sorted by organizational size (fig 9)

def sorter(x):
    if x == 'Small':
        return 1
    elif x == 'Medium':
        return 2
    else:
        return 3

violin = df
violin['sort'] = violin['size'].apply(sorter)
violin = violin.sort_values(by = 'sort')

20. A DataFrame was created to chart Large Organization data (fig 8)

Large = df.loc[df['size'] == 'Large']
Large = Large.groupby(['Brand_Generic','Parent_Organization'])[['Lives']].sum()
Large = Large.reset_index()

21. A DataFrame was created for a ski slope chart

ski_slope = df[['CONTRACT_ID','lowest_price_Q3','Lives','size','Brand_Generic']]
ski_slope = ski_slope.sort_values(by='lowest_price_Q3').reset_index(drop = True)

Proprietary database terms have been anonymized
ski_slope['lives_sum'] = ski_slope['Lives'].cumsum()

ski_slope['percent'] = ski_slope.lives_sum/ski_slope.Lives.sum()