DataForge

Last Update as of 12/06

Analysis of Amtrak's Data

This proposal contains a detailed description and plan of the data processing, insights & findings of Amtrak's recent data.

Created By:

Pranav Praveen Nair

Emil George Mathew

Abhishek Bhosale

Henry Kangten

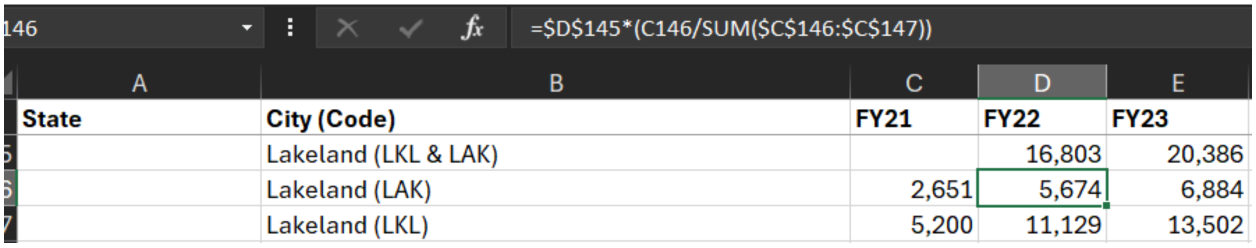
DATA PROCESSING

Amtrak data fetched from the below link, provided on Canvas
https://umd.instructure.com/courses/1368423/modules/items/13221176

Python script is used to transform the data into the format as per the ERD. Before reading the input data into python, certain transformations are carried out to ensure data consistency.

Excel Pre-processing

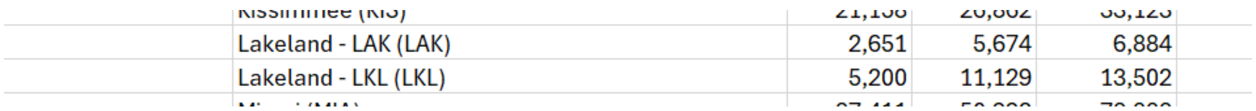
- There are two stations in Lakeland - the overall ridership for both is given for 2022 & 2023. For both these stations, the individual value is estimated in terms of the proportion of the 2021 values.



```
=$D$145*(C146/SUM($C$146:$C$147))
```

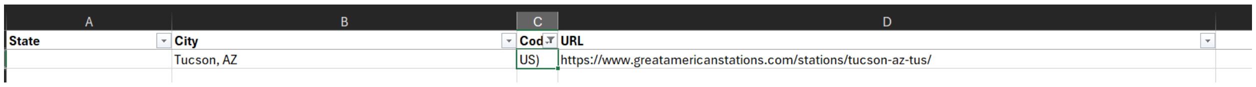
State	City (Code)	FY21	FY22	FY23
	Lakeland (LKL & LAK)		16,803	20,386
	Lakeland (LAK)	2,651	5,674	6,884
	Lakeland (LKL)	5,200	11,129	13,502

- The code is added to the station name, else there would be two stations with the same name.



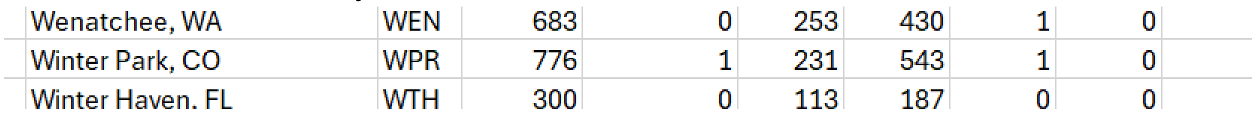
	Lakeland - LAK (LAK)	2,651	5,674	6,884
	Lakeland - LKL (LKL)	5,200	11,129	13,502

- The station code for Tucson is incorrect. It is manually changed in the input file.



State	City	Cod	URL
	Tucson, AZ	US)	https://www.greatamericanstations.com/stations/tucson-az-tus/

- The station code for Winter Park, CO was incorrect in the Budget tab. It has been corrected manually.



Wenatchee, WA	WEN	683	0	253	430	1	0
Winter Park, CO	WPR	776	1	231	543	1	0
Winter Haven, FL	WTH	300	0	113	187	0	0

Python Script

Below is the python script used to transform the data. Relevant comments are added for code readability.

#Import pandas packages & read data
import pandas as pd
import numpy as np
df = pd.read_excel("C:/Users/prana/Desktop/DBMS/Project/Amtrak-Original File.xlsx",sheet_name=None)

#Create State dataframe
State = pd.DataFrame(df['Station']) #main station tab

State['stateCode'] = State['City'].astype(str).str.split(',').str[1].str[:3] #fetch state code from city after splitting
State = State.drop_duplicates(['State','stateCode']).loc[:,['State','stateCode']] #drop state name duplicates
State = State[State['State'].isna() ==False] #filter rows with no state values

State['State'] = State['State'].str.rstrip().str.lstrip().drop_duplicates() #remove white spaces
State['stateCode'] = State['stateCode'].str.lstrip().str.rstrip() #remove white spaces
State = State[['stateCode','State']] #re-arrange columns
State.columns = ['stateCode','stateName'] #rename columns

#Clean original dataframe
df['Station']['State'] = df['Station']['State'].fillna(method='ffill') #fillNA values
df['Station']['State'] = df['Station']['State'].str.lstrip().str.rstrip() #remove white spaces
df['Station']['City'] = df['Station']['City'].str.lstrip().str.rstrip()
df['Station'].head()

df['Budget (in$K)']['Station Name'] = df['Budget (in$K)']['Station Name'].str.lstrip().str.rstrip() #remove white spaces
df['Budget (in$K)']['stationStateCode'] = df['Budget (in$K)']['Station Name'].str[-2:] #fetch state code from station name
df['Budget (in$K)']['Plan Year'] = df['Budget (in$K)']['Plan Year'].str.replace("FY","20").str.replace('PP','2024') #replace with year numbers

```
df['Budget (in$K)']['Budget'] = df['Budget (in$K)']['Budget'].astype(int)*1000  #multiply
with 1000 to get actual dollar amount
df['Budget (in$K)'].head()

df['Ridership']['State'] = df['Ridership']['State'].fillna(method='ffill').str.lstrip().str.rstrip()
#fillna state values
df['Ridership'] = df['Ridership'] [ df['Ridership']['City (Code) '].isna() == False ]  #remove
N/As since we only want to fetch station level values
df['Ridership'] = df['Ridership'][ df['Ridership']['City (Code)
'].astype(str).str.contains('Total') == False ]  #drop data with total in the string
df['Ridership']['City (Code) '] = df['Ridership'] ['City (Code) '].str.lstrip().str.rstrip()
#remove white spaces
df['Ridership']['Code'] = df['Ridership']['City (Code) '].astype(str).str.extract(r'\((\w{3})\)')
#extract 3 letter code
df['Ridership']['City (Code) '] = df['Ridership']['City (Code) '].str[:-6]  #remove city code
df['Ridership'] = pd.merge(df['Ridership'],State,left_on = 'State',right_on =
'stateName',how = 'left')  #fetch state key
df['Ridership'] ['City (Code) '] = df['Ridership'] ['City (Code) '] + ', ' + df['Ridership']
['stateCode']  #concat state key to station (LAK)
df['Ridership'] = df['Ridership'][['State','stateCode','City (Code)
','Code','FY21','FY22','FY23']] #rearrange columns
df['Ridership'].head()

df['Procurement'] = df['Procurement'] [ df['Procurement']['State'].isna() == False ]
#remove N/As since we only want to fetch state level procurement value
df['Procurement']['State'] = df['Procurement']['State'].str.lstrip().str.rstrip()  #remove white
spaces
df['Procurement'] = df['Procurement'].loc[:,['State','FY21','FY22','FY23']]  #drop station
column
df['Procurement'].head()

df['On-Time Performance (OTP)'] = df['On-Time Performance (OTP)']
[['Type','Route','Stations','FY21 OTP','FY22 OTP','FY23 OTP']]
updated_df = []
for index, row in df['On-Time Performance (OTP)'].iterrows(): #iterate through each row
    stations = row['Stations'].split('-')  #split stations column based on the hyphen
    for station in stations:  #run for loop to ensure rest of the data remains and append
to updated df
        updated_df.append({
          "Type": row['Type'],
         "Route": row['Route'],
         "Station": station.strip(),
         "FY21 OTP": row['FY21 OTP'],
         "FY22 OTP": row['FY22 OTP'],
         "FY23 OTP": row['FY23 OTP']
        })

df['On-Time Performance (OTP)'] = pd.DataFrame(updated_df)

df['On-Time Performance (OTP)']['Station'] = df['On-Time Performance
(OTP)']['Station'].str.lstrip().str.rstrip()  #remove white spaces
df['On-Time Performance (OTP)']['Route'] = df['On-Time Performance
(OTP)']['Route'].str.lstrip().str.rstrip()
df['On-Time Performance (OTP)']['Type'] = df['On-Time Performance
(OTP)']['Type'].str.lstrip().str.rstrip()
df['On-Time Performance (OTP)'] = df['On-Time Performance
(OTP)'].drop_duplicates(['Type','Route','Station']) #drop duplicates because of same
data for diffferent states


df['Employment'] = df['Employment'][df['Employment']['State'].isna() == False]
df['Employment']['State'] = df['Employment']['State'].str.lstrip().str.rstrip()  #remove white
spaces
df['Employment'].head()


#Create Stations dataframe
Station = pd.DataFrame(df['Station'].loc[:,['City','Code','State']])  #main stations tab
Station = pd.merge(Station,State,left_on = 'State',right_on = 'stateName',how = 'left')
#fetch state key
Station = Station.loc[:,['Code','City','stateCode']]
Station.head()

budgetStation = pd.DataFrame(df['Budget (in$K)']).loc[:,['Code','Station
Name','stationStateCode']]
budgetStation.columns = ['Code','City','stateCode']
budgetStation.head()

ridershipStation = pd.DataFrame(df['Ridership']).loc[:,['Code','City (Code) ','stateCode']]
ridershipStation.columns = ['Code','City','stateCode']
ridershipStation.head()

Station = pd.concat([Station,budgetStation,ridershipStation],axis = 0).drop_duplicates()
#fetch stations from other tabs that are not there on stations
Station.head()

Station.columns = ['stationCode','stationName','stateCode']
Station = Station[Station['stationName'].isna() == False]
```

#Create Budget Dataframe

```
Budget = pd.DataFrame(df['Budget (in$K)'])

Budget = Budget.drop_duplicates(['Type','Plan Year']).loc[:,['Type','Plan Year','Budget']]
#drop duplicates
Budget.columns = ['budgetType','budgetPlanYear','budgetTotal']  #rename cols
Budget['budgetYearID'] = [np.random.randint(100, 1000) for _ in range(len(Budget))]
#assign ID for each budget type & plan year combo
Budget = Budget[['budgetYearID','budgetType','budgetPlanYear','budgetTotal']]
#rearrange columns
```

#Create AllocatedBudget Dataframe

```
AllocatedBudget = pd.DataFrame(df['Budget (in$K)'].drop(['Station
Name','stationStateCode','Budget'],axis = 1))  #drop stationname, statecode,budget cols
AllocatedBudget.columns =
['budgetType','budgetPlanYear','stationCode','FY22','FY23','FY24','FY25','FY26']
#rename cols
AllocatedBudget = pd.melt(AllocatedBudget,id_vars =
['budgetType','budgetPlanYear','stationCode'], value_vars =
['FY22','FY23','FY24','FY25','FY26'],
                    value_name = 'allocatedBudget',var_name = 'allocatedBudgetYear')
#unpivot table

AllocatedBudget['allocatedBudgetYear'] =
AllocatedBudget['allocatedBudgetYear'].str.replace('FY','20') #replace FY with 20
AllocatedBudget['allocatedBudget'] = AllocatedBudget['allocatedBudget']*1000 #multiple
with 1000 to get the actual budget amount
```

```
#Create Route DataFrame
Route = pd.DataFrame(df['On-Time Performance (OTP)'])

Route = Route[['Type','Route']].drop_duplicates(['Type','Route'])  #drop duplicates
Route.columns = ['routeType','routeName']  #rename columns
unique_numbers = {name: f"{i:03}" for i, name in enumerate(Route['routeName'],
start=100)}
Route['routeID'] =Route['routeName'].map(unique_numbers) #assign random numbers
as identifiers for rout
Route = Route[['routeID','routeName','routeType']]  #rearrange cols


#Create OnTimePerformance DataFrame
OnTimePerformance= pd.DataFrame(df['On-Time Performance (OTP)']) #create otp df
OnTimePerformance = OnTimePerformance[['Route','Station','FY21 OTP','FY22
OTP','FY23 OTP']]

OnTimePerformance = pd.melt(OnTimePerformance, id_vars =
['Route','Station'],value_vars = ['FY21 OTP','FY22 OTP','FY23 OTP'],
                                        var_name = 'otpYear',value_name = 'otpValue')  #unpivot
table
OnTimePerformance['otpYear'] =
OnTimePerformance['otpYear'].str.replace('OTP','').str.replace('FY','20')  #rename
values of year

#clean the station names manually & map it with the correct station names; in case of
multiple stations in a city one of them is considered
OTPstations = OnTimePerformance ['Station'].drop_duplicates()
with pd.ExcelWriter('OTP-Station.xlsx') as writer:
   OTPstations.to_excel(writer, sheet_name='Stations', index=False)  #Write State to
'State' sheet

OTPstationsmap = pd.read_excel("C:/Users/prana/Desktop/DBMS/Project/OTP-Station-
map.xlsx",sheet_name=None) #manual map file
OnTimePerformance = pd.merge(OnTimePerformance,OTPstationsmap['Stations'],how
= 'left')  #left join to get station name
OnTimePerformance = OnTimePerformance.dropna(subset = ['New Station'])  #drop if
there are N/As
OnTimePerformance = OnTimePerformance[['Route','New Station','otpYear','otpValue']]
OnTimePerformance.columns = ['route','station','otpYear','otpValue']  #rename cols
```

```
OnTimePerformance = OnTimePerformance.drop_duplicates(['route','station','otpYear'])
#drop duplicates (same station appearing twice)


#Create StationMetrics DataFrame
StationMetrics = pd.DataFrame(df['Ridership'])

StationMetrics = StationMetrics.loc[:,['City (Code) ','FY21','FY22','FY23']]
StationMetrics = pd.melt(StationMetrics,id_vars = 'City (Code) ',value_vars =
['FY21','FY22','FY23'],var_name = 'year',value_name = 'metric')  #unpivot
StationMetrics['stationMetric'] = 'Ridership'  #create metric column
StationMetrics.columns =
['station','stationMetricYear','stationMetricValue','stationMetric']  #rename columns
StationMetrics.stationMetricValue = StationMetrics.stationMetricValue.fillna(0)  #fill nulls
with zero
StationMetrics['stationMetricYear'] =
StationMetrics['stationMetricYear'].str.replace('FY','20')  #replace year values


#Create StateMetrics DataFrame
Employment = pd.DataFrame( df['Employment'].loc[:,['State','FY21','FY22','FY23'] ] )
EmploymentWage = pd.DataFrame(
df['Employment'].loc[:,['State','$FY21','$FY22','$FY23'] ] )

Employment = pd.melt(Employment,id_vars = 'State',value_vars =
['FY21','FY22','FY23'],value_name = 'stateMetricValue',var_name = 'stateMetricYear')
#unpivot table
Employment['stateMetric'] = 'Employment' #create metric name column

EmploymentWage = pd.melt(EmploymentWage,id_vars = 'State',value_vars =
['$FY21','$FY22','$FY23'],
                     value_name = 'stateMetricValue',var_name = 'stateMetricYear')
#unpivot table
EmploymentWage['stateMetric'] = 'EmploymentWage' #create metric name column

Procurement = pd.DataFrame(df['Procurement'])

Procurement = Procurement.loc[:,['State','FY21','FY22','FY23']]
Procurement = pd.melt(Procurement,id_vars = 'State',value_vars =
['FY21','FY22','FY23'],var_name = 'stateMetricYear',value_name = 'stateMetricValue')
#unpivot
Procurement['stateMetric'] = 'Procurement'  #create metric column
Procurement['stateMetricValue'] = Procurement['stateMetricValue'].fillna(0)  #fill NA
values with zero

StateMetrics = pd.concat([Employment,EmploymentWage,Procurement],axis = 0)
#concat all three tables
StateMetrics = StateMetrics[StateMetrics.State.isna() == False]  #remove rows where
state is null
StateMetrics['stateMetricYear'] =
StateMetrics['stateMetricYear'].str.replace('$','').str.replace('FY','20')  #replace years with
the numbers
```

#Join to fetch the foreign keys

```
AllocatedBudget = pd.merge(AllocatedBudget,Budget,how='left')  #fetch budgetYearID
AllocatedBudget =
AllocatedBudget.loc[:,['budgetYearID','stationCode','allocatedBudgetYear','allocatedBud
get']]
AllocatedBudget = AllocatedBudget[AllocatedBudget['allocatedBudget'] != "$              "]
#remove incorrect values


OnTimePerformance = pd.merge(OnTimePerformance,Route,left_on = 'route',right_on
= 'routeName',how='left')  #fetch routeID
OnTimePerformance = pd.merge(OnTimePerformance,Station,left_on =
'station',right_on = 'stationName',how = 'left') #fetch State code
OnTimePerformance =
OnTimePerformance.loc[:,['routeID','stationCode','otpYear','otpValue']]


StationMetrics = pd.merge(StationMetrics,Station,left_on = 'station',right_on =
'stationName',how='left')  #fetch stationID
StationMetrics =
StationMetrics.loc[:,['stationCode','stationMetric','stationMetricYear','stationMetricValue']]
#rearrange columns
```

```
StateMetrics = pd.merge(StateMetrics,State, left_on = 'State',right_on =
'stateName',how='left')  #fetch stationcode
StateMetrics =
StateMetrics.loc[:,['stateCode','stateMetric','stateMetricYear','stateMetricValue']]
#rearrange columns


#Modify stations dataframe
Station = Station.drop_duplicates('stationCode')  #drop duplicate station codes that
might appear since same code can have multiple names
Station['stationName'] = Station['stationName'].str.replace(r',.*?(–|$)', '', regex=True)
#reformat the stationName


#Export final cleaned version of data into excel
OnTimePerformance.head()

with pd.ExcelWriter('Amtrak-CleanedData-Final.xlsx') as writer:
   State.to_excel(writer, sheet_name='State', index=False)  # Write State to 'State'
sheet
   Station.to_excel(writer, sheet_name='Station', index=False)  # Write Station to
'Station' sheet
   StateMetrics.to_excel(writer, sheet_name='StateMetrics', index=False)  # Write
StateMetrics to 'StateMetrics' sheet
   StationMetrics.to_excel(writer, sheet_name='StationMetrics', index=False)  # Write
StationMetrics to 'StationMetrics' sheet
   Route.to_excel(writer, sheet_name='Route', index=False)  # Write Route to 'Route'
sheet
   OnTimePerformance.to_excel(writer, sheet_name='OnTimePerformance',
index=False)  # Write OnTimePerformance to 'OnTimePerformance' sheet
   Budget.to_excel(writer, sheet_name='Budget', index=False)  # Write Budget to
'Budget' sheet
   AllocatedBudget.to_excel(writer, sheet_name='AllocatedBudget', index=False)  #
Write BudgetValue to 'BudgetValue' sheet
```

References

1. State Fact Sheets | Amtrak
2. Reports & Documents | Amtrak
3. Train Routes | Amtrak
4. Wikipedia | Amtrak Stations
5. https://railroads.dot.gov/
6. https://www.amtrak.com/home.html
7. https://www.bts.gov/
8. https://www.transportation.gov/
9. https://www.amtrak.com/reports-documents

Findings, Insights and Recommendations

Findings:

1. **Yearly Budget Trends**:
   - The budget allocations across different categories (Construction, Deployment, Design) fluctuate over the years, with certain categories peaking in specific years. For instance, the "Construction" budget significantly peaked in 2022, signaling a period of increased focus on infrastructure development. Meanwhile, "Design" consistently received lower allocations, reflecting a relatively smaller investment in the design phase in comparison to other areas.
2. **Dominance of Construction**:
   - Over the period from 2020 to 2023, "Construction" budgets dominated, highlighting a strategic push towards infrastructure projects. This was particularly evident in 2020 and 2021, where the "Construction" category received the largest share of the budget, suggesting significant investments in building and physical infrastructure.
3. **Shifting Priorities**:
   - There were shifts in priorities over time, as seen in the "Deployment" budget, which peaked in 2019 but showed a noticeable decline in the following years. This may reflect a strategic shift, where focus transitioned from deployment and operational expenses to long-term construction and development projects.
4. **High OTP in the Northeast Corridor**:
   - The Northeast Corridor showed the highest On-Time Performance (OTP) at 0.8349, outperforming other regions. This indicates a successful model for operations that could be replicated in other regions to enhance overall performance.
5. **State-Supported Routes Perform Well**:
   - State-supported routes, while performing slightly below the Northeast Corridor, achieved a commendable OTP of 0.7872, indicating that collaboration with state authorities for route optimization is yielding positive results.
6. **Low OTP in Long-Distance Routes**:
   - Long-distance routes were identified as underperforming, with an OTP of only 0.4896. This indicates an urgent need for infrastructure investment

and operational improvements to increase punctuality and efficiency on these routes.

7. **Ridership Trends**:
    - New York leads with the highest ridership at 27.5 million, followed by California at 18 million. Other states, including Pennsylvania, D.C., and Illinois, contribute notably to overall ridership figures. The high ridership reflects key urban transit hubs and suggests areas of growth for expanding services.
8. **Procurement Spending Disparities**:
    - California has the highest procurement spending at over $1.35 billion, significantly outspending other states like New York ($1.34 billion) and Pennsylvania ($1.03 billion). This suggests that California is focusing heavily on procurement to sustain or expand its operations. In contrast, many central states show much lower procurement spending, potentially indicating underinvestment in operational resources.

Suggestions and recommendations:

1. **Review Allocation Rationale**:
    - It is essential to evaluate the rationale behind higher budget allocations for "Construction" in specific years, particularly during 2020–2023, to ensure that investments align with long-term strategic goals. Understanding the drivers behind these decisions can help optimize future spending and improve project outcomes.
2. **Balance Budget Distribution**:
    - Future budget allocations should aim for a more balanced distribution across "Construction," "Deployment," and "Design." Allocating resources evenly across these areas can enhance efficiency, reduce bottlenecks, and ensure that every phase of project development is adequately supported.
3. **Strategic Planning and Stabilization**:
    - Analyzing historical budget allocation trends can help stabilize future budget planning, reduce the risk of over-reliance on any single category, and allow for more flexible resource allocation based on project requirements.
4. **Improve Long-Distance Routes**:
    - Significant investment in infrastructure and operational improvements is needed to enhance OTP on long-distance routes. Focused investments in

    scheduling, maintenance, and station facilities could significantly reduce delays and improve service quality.

5. **Replicate Northeast Corridor Success**:
    - Best practices and successful operational strategies from the Northeast Corridor, which boasts the highest OTP, should be studied and applied to other regions. This could involve enhancing infrastructure, optimizing scheduling, or leveraging technology to improve punctuality.
6. **Enhance State Collaboration**:
    - Strengthening partnerships with state authorities for state-supported routes is essential. By working collaboratively, performance can be further optimized, and OTP for these routes could increase to match or even exceed the performance of the Northeast Corridor.
7. **Prioritize High-Ridership States**:
    - States like New York and California, which already have high ridership, should continue to receive investment to maintain and improve service quality. Additionally, strategies from these states could be adapted and applied to regions with lower ridership to stimulate growth.
8. **Promote Ridership in Low-Ridership States**:
    - States with lower ridership should focus on expanding services, improving infrastructure, and targeting key areas for growth. Analyzing the factors contributing to high ridership in top states and applying them to lower-performing regions could help boost ridership across the board.
9. **Investigate High Procurement States**:
    - The procurement spending patterns in high-investment states such as California, New York, and Pennsylvania should be analyzed. Identifying efficiencies or areas for cost savings can provide insights into how to optimize procurement in other regions.
10. **Reallocate Resources Strategically**:
    - Resources should be strategically reallocated to underfunded states that show potential for improvement or growth. By directing more funds to these regions, it is possible to enhance infrastructure, reduce delays, and improve service quality across a broader geographical area.
11. **Optimize Procurement Practices**:
    - Procurement strategies in high-performing states should be studied and used as a benchmark to streamline procurement processes in lower-spending regions. Implementing these best practices could help enhance procurement efficiency and optimize the impact of every dollar spent.