**NYC TAXI TRIPS**

This dataset was chosen because of the well-defined data dictionary and the richness of the data across multiple years. The total size of the dataset was too huge to be analyzed by Excel. PowerBI was an option, but I realized during aggregates would take a while, SQL queries are more efficient and faster, therefore, I decided to import the data to SQL Server. Therefore, the first step was to create a SQL database, create a table and insert the data. This provided a perfect opportunity to showcase SQL and PowerBI skills. It’s going to be challenging but fun.

Steps.

1. Study the data dictionary to understand the data types.

* The data types were identified based on the data dictionary definitions and observations from the data tables to place best match.
* Assumption: The description from the data dictionary table was correctly defined.

1. Create SQL Server database
2. Create a table NYC\_Trips and Taxi\_Zones

* Create tables for 2017, 2018, 2019 and 2020 using schemas for each year

1. Import the data

* Insert the data for each year.
* After importing, the next step was to convert the datatypes back to the original types using ALTER Statements.

1. Explore the data in SQL Server database

* Check for rows with NULL
* To check if any of the tables have a NULL value across all columns the following queries were used:

1. CHECK FOR NULLS FOR ALL COLUMNS IN TRIPS\_2017
2. CHECK FOR NULLS FOR ALL COLUMNS IN TRIPS\_2018
3. CHECK FOR NULLS FOR ALL COLUMNS IN TRIPS\_2019
4. CHECK FOR NULLS FOR ALL COLUMNS IN TRIPS\_2020

|  |  |  |
| --- | --- | --- |
| **SN. No:** | **Year** | **Columns with NULL** |
| 1 | 2017 | VendorID |
| RatecodeID |
| Passenger\_Count |
| Payment\_Type |
| Trip\_Type |
| Congestion\_Surchage |
| 2 | 2018 | VendorID |
| RatecodeID |
| Passenger\_Count |
| Payment\_Type |
| Trip\_Type |
| Congestion\_Surchage |
| 3 | 2019 | VendorID |
| RatecodeID |
| Passenger\_Count |
| Payment\_Type |
| Trip\_Type |
| Congestion\_Surchage |
| 4 | 2020 | VendorID |
| RatecodeID |
| Passenger\_Count |
| Payment\_Type |
| Trip\_Type |
| Congestion\_Surchage |

* Observation: All Tables had at least one column with NULL values and the same columns for each year had some NULL values  
  Conclusion: The data was erroneously captured for some of the trips for the same columns for each different year
* Solution

1. Passenger\_Count field should have at least a value of 1, based on the assumption that one passenger was at least on the ride. Default to 1
2. Payment\_Type – default this to 5 for Unknown.
3. Congestion\_Surchage – based on (a) above, default to 0 as it’s unknown where it ended or passed
4. VendorID - default to ‘Unknown’ and map it to 0
5. RatecodeID - default to ‘Unknown’ instead of 1 for Standard rate to preserve data integrity. Also, the number or records with NULL for this field are hundreds of thousands.
6. Create Dimension Tables from the data dictionary:

* The following fields were identified as tables that are dimension tables that should be extracted.

1. VendorID
2. store\_and\_fwd\_flag
3. RatecodeID
4. payment\_type
5. trip\_type
6. PULocationID
7. DOLocationID
8. Explore the data in SQL.
9. Count of trips
10. Total trip distance by vendor
11. Total trips by year, quarter, month
12. Count of tips
13. Count of tips across passenger count
14. Trips per vendor per year
15. Trips by year, quarter, month
16. Average total amount per vendor by year
17. Average passenger count by vendor
18. Average passenger count by year, quarter, month
19. Ranking of the payment types
20. Ranking of the payment types by year
21. Ranking of pick-up locations
22. Ranking of drop-off locations
23. Average rating by vendor
24. Average rating by trip-type

Business Objectives.

1. Analyze passenger trips per year – to understand trips pattern such quarter rankings, growth trend YoY, QoQ
2. Understand customer tipping trends – against no of passengers, distance covered, pickup location, drop-off location, payment type, trip type.
3. Analyze trip distance covered – check for trends against no. of passengers
4. Aggregate the data before importing it into PowerBI
5. Create data models

Challenges & Solutions

|  |  |  |
| --- | --- | --- |
| Serial No | Challenge | Solution |
| 1. | On Insert into Taxi\_Zones table, fields ‘Borough’, Zone\_Name’ & ‘Service\_Zone’ had limited characters as it was created as a VARCHAR only | Created an ALTER Query to increase the size of characters to 500 |
| 2. | Altering multiple columns gave an error | Created separate ALTER queries for each field. |
| 3. | Large data files to GitHub. | The data files were too large to be uploaded to GitHub, the workaround was to upload to OneDrive on this link: [NYC\_TRIPS](https://afzdxb-my.sharepoint.com/:f:/g/personal/innocent_kyalo_alfazance_com/EiNppSKimd9JiAdfNknRNN0B3dGRIW0c0ZiDgrwPUzzSIA?e=xFGI0y) |
| 4. | Bulk Import data to Trips\_2020 Table | Realized there was an additional column Trip\_ID that was causing erroneous insertion of data, dropped the column.  - On attempting drop, I got a constraint error. Resolved this by dropping the constraint & proceeded to drop the column. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Challenges & Solutions**

1. Import of Data

* The raw data for the Trips table was voluminous to be inserted into the table using.
* Workaround 1
  + At first, I tried an BULK insert query which kept on giving errors about a mismatch between my table data type and my CSV data types. The initial BULK insert is in the SQL Queries folder named ‘Bulk Insert Trips Data Query’
* Workaround 2
  + After no success with workaround 1, I researched a technique that proved to work. It involved altering my existing fields data type to VARCHAR using query ‘ALTER TRIPS TABLE \_ ALL FIELDS TO VARCHAR’, importing the data and finally converting them back into the original data types.

1. Assuming all Trips Data has the same data fields.

* Initially it was assumed that all data files have the same number of data fields, and it was ordered in the same format. Upon investigation, I discovered that this was not consistent for all the years.
* Next Step: Create separate tables for each year and import the data separately to avoid inconsistencies.