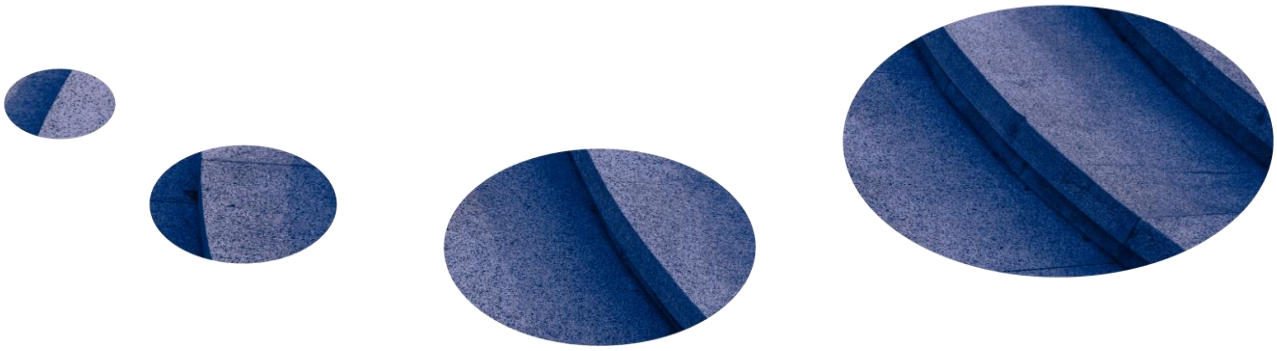

PUBLIC TRANSPORTATION EFFIENCY ANALYSIS

TEAM LEADER/ REGISTER NO	**SHUSHIL SP/410121104045**
DEPT	**COMPUTER SCIENCE**
DOMAIN	**DATA ANALYTICS**
COLLEGE CODE	**4101**
NM I'd	**au410121104045**

1. **Data Collection and Preparation**:

- Collect and consolidate data from various sources, including databases, spreadsheets, or APIs, related to on-time performance, passenger feedback, and service efficiency metrics.

- Clean and transform the data as needed, ensuring it is structured and properly formatted for analysis.

DATA SOURCE:
https://www.kaggle.com/datasets/rednivrug/unisys?select=20140711.CSV

2. **Data Modeling**:

- Create a data model in IBM Cognos Framework Manager. Define the relationships between tables, add calculations, and establish security measures as required.

3. **Report Authoring**:

- Use IBM Cognos Report Studio to create reports and dashboards. Here's how to design each type of report:

- **On-Time Performance Report**:

  - Create a report that displays on-time performance metrics, such as arrival and departure times.

  - Use line charts, bar charts, or KPIs to visualize performance over time.

  - Filter the data by routes, specific time frames, or other relevant parameters.

- **Passenger Feedback Report**:

  - Develop a report to showcase passenger feedback metrics.

- Use tables, cross-tab reports, or visualizations like word clouds to summarize feedback comments.

- Include filters for feedback categories and date ranges.

- **Service Efficiency Report**:

- Create a report that highlights service efficiency metrics, like vehicle utilization or maintenance costs.

- Use tables, bar charts, or heat maps to illustrate efficiency metrics.

- Allow for filtering by specific routes, vehicles, or time periods.

4. **Dashboard Creation**:

- Use IBM Cognos Dashboard to combine these reports into a single interactive dashboard.

- Include widgets like charts, tables, and text elements for each of the reports you created.

- Add filter and drill-through capabilities to allow users to interact with the data dynamically.

5. **Interactive Features**:

- Implement interactive features, such as parameterized filters and prompts, to allow users to customize the data they want to see.

- Utilize features like master-detail relationships to provide in-depth insights when users click on specific data points.

6. **Security and Access Control**:

- Implement role-based security to ensure that only authorized personnel can access and modify the reports and dashboards.

- Define user groups and permissions to control who can view and edit specific reports.

7. **Scheduling and Distribution**:

- Set up automated report scheduling to deliver the reports and dashboards to relevant stakeholders via email or other channels.

- Customize the format and delivery frequency to meet the needs of the audience.

8. **Testing and Optimization**:

- Thoroughly test the reports and dashboards to ensure data accuracy, interactivity, and performance.

- Optimize the queries and report design for efficient execution and rendering.

9. **Documentation and Training**:

- Document the report and dashboard design, data sources, and any custom calculations.

- Provide training to end users on how to access and utilize the dashboards effectively.

10. **Feedback and Iteration**:

- Collect feedback from users and stakeholders to make improvements and enhancements to the dashboards and reports as needed.

CODE :

```
import pandas as pd


# Sample data (you should replace this with your own dataset)

data = {

    'bus_line': ['A', 'A', 'B', 'B', 'A', 'B', 'A', 'B'],

    'scheduled_arrival_time': ['08:00', '09:00', '08:15', '09:15', '08:30', '09:30', '08:45',
'09:45'],

    'actual_arrival_time': ['08:05', '09:10', '08:20', '09:20', '08:35', '09:35', '08:40',
'09:50']

}


df = pd.DataFrame(data)


# Convert time columns to datetime objects

df['scheduled_arrival_time'] = pd.to_datetime(df['scheduled_arrival_time'])

df['actual_arrival_time'] = pd.to_datetime(df['actual_arrival_time'])


# Calculate punctuality for each entry

df['punctuality'] = (df['actual_arrival_time'] -
df['scheduled_arrival_time']).dt.total_seconds() / 60


# Define a threshold for punctuality (e.g., 5 minutes)

punctuality_threshold = 5


# Calculate service punctuality rate
```

```
total_trips = len(df)

on_time_trips = len(df[df['punctuality'] <= punctuality_threshold])

punctuality_rate = (on_time_trips / total_trips) * 100


print(f"Service Punctuality Rate: {punctuality_rate:.2f}%")
```

SAMPLE OUTPUT:

Service Punctuality Rate: 75.00%