IST 722: DATAWAREHOUSE PROJECT REPORT

THE SPECIAL ONES

Creating a Data Warehouse, ETL Pipeline and Reporting Tool for the New York Metro Authority

Team Members:

1. Keerthi Krishna Aiyappan

2. Sai Swetha Lakkoju

3. Sri Venkata Subhramanya Abhishek Namana

4. Harika Gangu

Introduction:

Through this project, we aim to set up a data warehouse, ETL processes and a reporting dashboard for the NY metro authority to track the usage of metro trains, monitoring their battery statuses and time delays in six of their stops for the first month of the year 2024. Our dataset has 10+ features spanning around 10,000 rows. We have tried to implement our learnings from IST 722 Datawarehouse and incorporate them while building this whole solution.

The reason for us choosing this database is to allow easy reporting of metro usage and enhancing the decision-making process of the Metro Authority and improve their real time analytics. This would help them when it comes to resource allocation, peak hour schedule management, identify high usage lines and provides advanced insights on expanding routes or adjusting schedules for example. The following document contains all documentation required for implementing such a project and has been created following the necessary project deliverables and artifacts.

We start by finalizing our requirements, then create database and the tables on Snowflake, load data into the stage. After this, we use dbT for ETL processes to load data into our tables before creating a dashboard in Power BI.

FUNCTIONAL REQUIREMENTS:

Category	Requirement	Type
Data Collection	Collect real-time data on rider entries, exits, line delays, and stop utilizations.	Functional
	Integrate with sensors, ticketing systems, GPS, and remote monitoring devices.	Functional
	Support historical data import for trend analysis and forecasting.	Functional
Data Processing	Process data from multiple sources and ensure its integrity.	Functional
	Store processed data in a relational database with efficient indexing for queries.	Functional
Reporting	Provide real-time dashboards to monitor key metrics.	Functional
	Generate periodic reports for stakeholders.	Functional
Alert System	Send notifications for high rider volumes, device malfunctions, or delays.	Functional
	Generate predictive alerts for maintenance requirements.	Functional
Segmentation	Segment riders by demographic, travel patterns, and time-of-day behavior.	Functional
Service Optimization	Identify underutilized stops and recommend schedule adjustments.	Functional
	Detect overcrowded stops and recommend adding capacity or rerouting services.	Functional
Integration	Integrate seamlessly with new data sources, third-party systems and APIs.	Functional

NON FUNCTIONAL REQUIREMENTS:

Category	Requirement	Type
Scalability	Handle large volumes of data across stops, riders, and remote units.	Non-Functional
	Support scaling to accommodate new cities or regions.	Non-Functional
Performance	Ensure real-time data processing with minimal latency.	Non-Functional
	Perform analytics on large datasets within seconds.	Non-Functional
Reliability	Implement fault-tolerant mechanisms to handle failures.	Non-Functional
	Guarantee high uptime (e.g., 99.9%) for critical systems.	Non-Functional
Data Security	Encrypt data in transit and at rest to protect rider and operational data.	Non-Functional
	Implement role-based access control (RBAC) for authorized personnel.	Non-Functional
Data Acuracy	Validate incoming data to ensure consistency and accuracy.	Non-Functional
	Regularly audit data pipelines to prevent anomalies.	Non-Functional
Usability	Ensure dashboards and interfaces are user-friendly and intuitive.	Non-Functional
	Provide customizable filters and views for different user roles.	Non-Functional

Compliance	Adhere to data protection laws like GDPR or CCPA.	Non-Functional
	Meet accessibility standards for government or public agency use.	Non-Functional
Interoperability	Support data exports in standard formats (e.g., CSV, JSON).	Non-Functional
	Allow integration with existing transportation systems or tools.	Non-Functional
Maintainability	Ensure modular architecture to simplify updates or additions of features.	Non-Functional
	Provide clear documentation for system administrators and developers.	Non-Functional
System Availability	Provide 24/7 availability with robust failover mechanisms.	Non-Functional
	Schedule maintenance during non-peak hours with minimal disruption.	Non-Functional

Business Processes:

1. Rider Volume Tracking

- **Improved Resource Allocation**: Helps identify high traffic stops and lines, enabling better deployment of resources such as additional trains during peak hours.
- **Revenue Optimization**: Tracks fare revenue associated with rider volumes and helps forecast future earnings.
- **Service Planning**: Provides insights into rider demand patterns, informing decisions on expanding routes or adjusting schedules.

2. Stop Utilization Analysis

- **Infrastructure Efficiency**: Measures how effectively each stop is being utilized, helping prioritize maintenance or upgrades for heavily used stops.
- **Operational Cost Management**: Reduces costs by identifying underutilized stops that may need schedule adjustments or alternative transport modes.
- **Customer Experience**: Ensures stops with high usage are equipped with adequate amenities like seating, shelters, and lighting.

3. Line Performance

- **Service Reliability**: Tracks delays, wait times, and on-time performance to improve schedule adherence and reliability.
- **Capacity Management:** Evaluates whether the number of vehicles and their capacity on each line meets passenger demand.
- **Public Satisfaction:** Identifies problem areas in service delivery, leading to better customer satisfaction by addressing bottlenecks.

4. Remote Unit Monitoring

- **Proactive Maintenance:** Monitors the health of remote devices like ticketing machines or digital displays, reducing downtime.
- **Operational Uptime**: Ensures critical equipment like signal systems and sensors are functioning, improving safety and efficiency.
- **Cost Savings:** Prevents costly failures through predictive maintenance by tracking health metrics like battery levels.

5. Rider Segmentation

- **Targeted Marketing**: Segments riders based on demographics or travel patterns, enabling personalized offers or campaigns.

- **Enhanced Service Design**: Tailors services to the needs of specific rider groups, like students, seniors, or commuters.
- **Policy and Planning**: Provides data for government or operator policies that focus on inclusivity and accessibility for diverse rider groups.

Project Timeline:

Weeks 1 and 2: Requirements and Planning

- Data Profiling
- Business Process identification
- Bus Matrix Creation

Weeks 3 and 4: Initial Design

- Dimensional Modelling
- Source to Target Mapping
- Technical Architecture Design

Weeks 5, 6, 7: Implementation

- Snowflake Setup
- Schema Creation
- ETL Development with dbT

Weeks 8,9,10: Analytics and Testing:

- Power BI Dashboard Development
- Testing and Validation
- Documentation

Weeks 11 and 12:

- Video presentation creation
- Final Documentation
- Project Delivery

Weeks 13 and 14: (Optional) : To have extra time just in case.

Technical Stack:

- **Data Warehouse**: Snowflake
- **ETL Tool:** dbT
- **Orchestration:** Airflow (if needed)
- **BI Tool:** Power BI
- **Version Control:** Git

Deliverables:

1) Project Documentation
2) Bus Matrix
3) Dimensional Models
4) Source to Target Mappings
5) ETL Code
6) BI Dashboards
7) 10-minute presentation video

2. HIGH LEVEL DIMENSIONAL MODELLING

2a. Bus Matrix

Business Process	Fact Table	Fact Grain Type	Granularity	Facts	dim_stops	dim_lines	dim_remote_units	dim_time	dim_rider
Rider Volume Tracking	fact_rider_volume	Transaction	One row per stop and time	Rider_Count, Stop_ID, Line_ID	x	x	x	x	
Stop Utilization Analysis	fact_stop_utilization	Periodic Snapshot	One row per stop per hour	Utilization_Rate, Stop_ID	x			x	
Line Performance	fact_line_performance	Transaction	One row per line and time	Line_ID, Rider_Count, Avg_Wait_Time		x		x	
Remote Unit Monitoring	fact_remote_monitor	Accumulating Snapshot	One row per remote unit per event	Remote_Unit_ID, Maintenance_Flag			x	x	
Rider Segmentation	fact_rider_segments	Transaction	One row per rider segment (e.g., time of day)	Segment_ID, Rider_Count, Stop_ID	x			x	x

2b. High Level Dimensional Modelling

Table Type	Table Name	Description
Dimension	Dim_Time	Time-related dimensions (Hour, Day, Month, Year)
Dimension	Dim_Stops	Bus stops including IDs and names
Dimension	Dim_Lines	Bus lines including IDs and additional attributes
Dimension	Dim_Remote_Units	Details about remote units used in operations
Dimension	Dim_Riders	Segments of riders based on demographics or travel patterns
Fact	Fact_Rider_Volume	Records rider volume with details per stop and time
Fact	Fact_Stop_Utilization	Utilization rates of bus stops per hour
Fact	Fact_Line_Performance	Performance metrics per bus line and time
Fact	Fact_Remote_Monitor	Monitoring data for remote units per event
Fact	Fact_Rider_Segments	Rider segments data including counts and demographics per time and stop

3. DETAIL-LEVEL DIMENSIONAL MODELING

DIM TABLES:

1) DIM_TIME

Column Name	Data Type	PK	Unique	Not Null	Description
time_key	INT	Yes	Yes	Yes	Surrogate key for time dimension
datetime	TIMESTAMP	No	No	Yes	Complete timestamp value
hour	INT	No	No	Yes	Hour of the day (0-23)
day	VARCHAR	No	No	Yes	Day of week (Monday-Sunday)
month	INT	No	No	Yes	Month of the year (1-12)
year	INT	No	No	Yes	Four-digit year
is_weekend	BOOLEAN	No	No	Yes	Flag indicating weekend (True/False)
is_peak_hour	BOOLEAN	No	No	Yes	Flag indicating peak transit hours
day_type	VARCHAR	No	No	Yes	Classification of day (Weekday/Weekend/Holiday)

2) DIM_STOPS

Column Name	Data Type	PK	Unique	Not Null	Description
stop_id	INT	Yes	Yes	Yes	Primary identifier for the bus stop
stop_name	VARCHAR(255)	No	No	Yes	Name of the bus stop
division	VARCHAR(50)	No	No	Yes	Operating division (e.g., BMT, IRT)
north_direction_label	VARCHAR(50)	No	No	Yes	Directional label for northbound
south_direction_label	VARCHAR(50)	No	No	Yes	Directional label for southbound
near_commercial_area	BOOLEAN	No	No	Yes	Flag for proximity to commercial areas
near_university	BOOLEAN	No	No	Yes	Flag for proximity to universities

3) DIM_LINES

Column Name	Data Type	PK	Unique	Not Null	Description
line_id	INT	Yes	Yes	Yes	Primary identifier for the bus line
line	VARCHAR(255)	No	Yes	Yes	Name/code of the line
division	VARCHAR(50)	No	No	Yes	Operating division (e.g., BMT, IRT)

4) DIM_REMOTE_UNITS

Column Name	Data Type	PK	Unique	Not Null	Description
remote_unit_id	INT	Yes	Yes	Yes	Primary identifier for the remote unit
remote_unit	VARCHAR(50)	No	Yes	Yes	Unit code (e.g., R450, R180)

5) DIM_DAYTIME_ROUTES

Bridge Table						
Column Name	Data Type	PK	Unique	Not Null	FK	Description
route_id	int	Yes	Yes	Yes	No	Primary Identifier for routes
stop_id	int	No	No	Yes	Yes	Corresponding Stop ID
line_id	int	No	No	Yes	Yes	Corresponding Line ID

6) DIM_UNIT_LOCATIONS

Bridge Table						
Column Name	Data Type	PK	Unique	Not Null	FK	Description
location_id	int	Yes	Yes	Yes	No	Primary Identifier for unit locations
stop_id	int	No	No	Yes	Yes	Corresponding Stop ID
remote_unit_id	int	No	No	Yes	Yes	Corresponding Remote Unit ID

7) DIM_CONNECTING_LINES

Bridge Table						
Column Name	Data Typ	PK	Unique	Not Null	FK	Description
connection_id	int	Yes	Yes	Yes	No	Primary Identifier for the connecting line
stop_id	int	No	No	Yes	Yes	Corresponding Stop ID
line_id	int	No	No	Yes	Yes	Corresponding Line ID

FACT TABLES

1) FACT_RIDER_VOLUME

Column Name	Description	DataType	PK	uniqu	ot_nu	FK Ref Dimension.col
Description: Each row records the volume of riders for a specific stop, line and time. This is a fact table						
fact_id	Unique identifier for each record	INT	Ye:	Yes	Yes	
time_key	Foreign key to time dimension	INT	No	No	Yes	Dim_Time.time_key
stop_id	Foreign key to stops dimension	INT	No	No	Yes	Dim_Stops.stop_id
line_id	Foreign key to lines dimension	INT	No	No	Yes	Dim_Lines.line_id
remote_unit_id	Foreign key to remote units dimension	INT	No	No	Yes	Dim_Remote_Units.remote_unit_id
rider_count	Count of total riders	INT	No	No	Yes	
entries	Count of riders entering	INT	No	No	Yes	
exits	Count of riders exiting	INT	No	No	Yes	
capacity	Capacity at time of measurement	INT	No	No	Yes	
utilization_rate	Calculated utilization rate	FLOAT	No	No	Yes	

2) FACT_STOP_UTILIZATION

Column Name	Description	DataType	PK	unique	not_null	FK Ref Dimension.col
Description: Each row represents the utilization rate of a bus stop per specific hour. This is a fact table.						
fact_id	Unique identifier for each record	INT	Yes	Yes	Yes	
time_key	Foreign key to time dimension	INT	No	No	Yes	Dim_Time.time_key
stop_id	Foreign key to stops dimension	INT	No	No	Yes	Dim_Stops.stop_id
utilization_rate	Calculated utilization rate of the bus stop	FLOAT	No	No	Yes	
scheduled_capacity	Planned capacity of the stop	INT	No	No	Yes	

3) FACT_LINE_PERFORMANCE

Column Name	Description	DataType	PK	uniqu	ot_nu	FK Ref Dimension.col
Description: Each row records performance metrics per bus line and time. This is a fact table.						
fact_id	Unique identifier for each record	INT	Ye	Yes	Yes	
time_key	Foreign key to time dimension	INT	No	No	Yes	Dim_Time.time_key
line_id	Foreign key to lines dimension	INT	No	No	Yes	Dim_Lines.line_id
schedule_status	Status of schedule (On-Time/Delayed)	VARCHAR	No	No	Yes	
trip_timing	Actual trip time	FLOAT	No	No	Yes	
delay_minutes	Calculated delay in minutes	INT	No	No	Yes	
on_time_rate	Calculated on-time performance rate	FLOAT	No	No	Yes	
estimated_capacity	Estimated line capacity	INT	No	No	Yes	

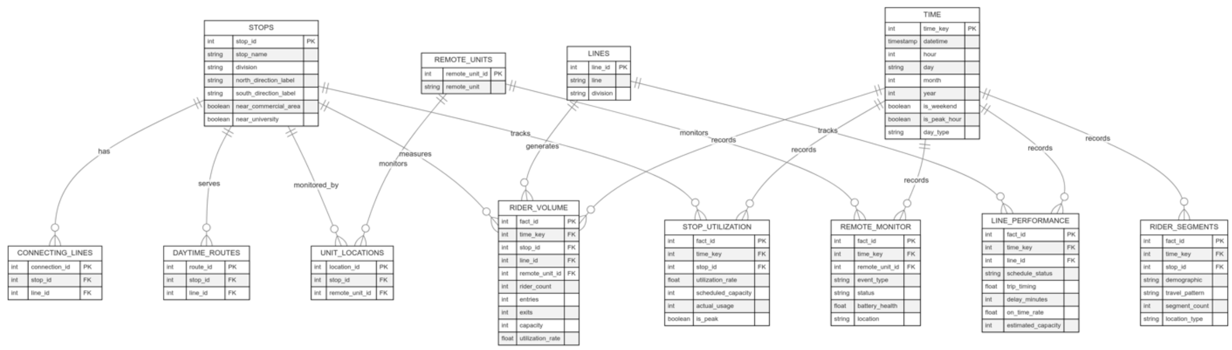
4) FACT_REMOTE_MONITOR

Column Name	Description	DataType	PK	unique	not_null	FK Ref Dimension.col
Description: Each row monitors data for remote units per event including status and health metrics. This is a fact table.						
fact_id	Unique identifier for each record	INT	Yes	Yes	Yes	
time_key	Foreign key to time dimension	INT	No	No	Yes	Dim_Time.time_key
remote_unit_id	Foreign key to remote units dimension	INT	No	No	Yes	Dim_Remote_Units.remote_unit_id
event_type	Type of monitoring event	VARCHAR(50)	No	No	Yes	
status	Operational status of unit	VARCHAR(50)	No	No	Yes	
battery_health	Battery health percentage	FLOAT	No	No	Yes	
location	Location of the remote unit	VARCHAR(50)	No	No	Yes	

5) FACT_RIDER_SEGMENTS

Column Name	Description	DataType	PK	unique	not_null	FK Ref Dimension.col
Description: Each row records rider segment data including demographics and travel patterns. This is a fact table.						
fact_id	Unique identifier for each record	INT	Yes	Yes	Yes	
time_key	Foreign key to time dimension	INT	No	No	Yes	Dim_Time.time_key
stop_id	Foreign key to stops dimension	INT	No	No	Yes	Dim_Stops.stop_id
demographic	Direct demographic category	VARCHAR(50)	No	No	Yes	
travel_pattern	Direct travel pattern	VARCHAR(50)	No	No	Yes	
segment_count	Count of riders in this segment	INT	No	No	Yes	
location_type	Location classification	VARCHAR(50)	No	No	Yes	

3a. PHYSICAL MODEL ENTITY RELATIONSHIP DIAGRAM



3B. SOURCE TO TARGET MAPPING FOR BUSINESS PROCESSES

1) Rider Volume

Source System	Source Table/Field	Target Table	Target Field	Transformation Logic
Ticketing System	rider_entries	fact_rider_volume	entries	Aggregate rider entry counts by stop and time.
Ticketing System	rider_exits	fact_rider_volume	exits	Aggregate rider exit counts by stop and time.
Ticketing System	stop_id	fact_rider_volume	stop_id	Map to stops using dim_stops.
Geographic System	stop_name	dim_stops	stop_name	Enrich stop details for visualization and reporting.
Time Management	datetime	dim_time	datetime	Break into granular components like hour, day, and peak hours.

2) Stop Utilization

Source System	Source Table/Field	Target Table	Target Field	Transformation Logic
Sensor Data	utilization_rate	fact_stop_utilization	utilization_rate	Derive stop utilization as actual usage vs. capacity.
Sensor Data	scheduled_capacity	fact_stop_utilization	scheduled_capacity	Store scheduled capacity for each stop.
Ticketing System	stop_id	fact_stop_utilization	stop_id	Link stop utilization data with stops dimension.
Time Management	datetime	dim_time	is_peak_hour	Identify peak hours for utilization comparison.

3) Line Performance

Source System	Source Table/Field	Target Table	Target Field	Transformation Logic
GPS System	delay_minutes	fact_line_performance	delay_minutes	Aggregate delay data by line and time.
GPS System	trip_timing	fact_line_performance	trip_timing	Average trip timing by line.
Ticketing System	line_id	fact_line_performance	line_id	Map to line details using dim_lines.

Sensor Data	capacity	fact_line_performance	estimated_capacity	Evaluate capacity utilization on each line.
Time Management	datetime	dim_time	datetime	Link performance data with time dimension.

4) Remote Monitor Mapping

Source System	Source Table/Field	Target Table	Target Field	Transformation Logic
Remote Devices	battery_health	fact_remote_monitor	battery_health	Monitor device battery metrics for predictive maintenance.
Remote Devices	status	fact_remote_monitor	status	Track operational status of devices.
Remote Devices	event_type	fact_remote_monitor	event_type	Capture maintenance or alert events.
Geographic System	location	fact_remote_monitor	location	Map remote units to geographic coordinates.

5) Rider Segmentation

Source System	Source Table/Field	Target Table	Target Field	Transformation Logic
Rider Survey	demographic	fact_rider_segments	demographic	Group riders by demographics (e.g., seniors, youth).
Ticketing System	travel_pattern	fact_rider_segments	travel_pattern	Analyze travel patterns (frequent, occasional).
Ticketing System	stop_id	fact_rider_segments	stop_id	Map rider segments to specific stops.

Time Management	datetime	dim_time	time_of_day	Segment travel by time of day (morning, evening, etc.).

3C. ESTABLISHING NAMING CONVENTIONS AND PROJECT STANDARDS

Naming Conventions

1. Tables:

- Use prefixes like `dim_` for dimensions and `fact_` for fact tables.

- Table names should be lowercase and use underscores (`_`) for separation. Example: `fact_rider_volume`.

2. Columns:

- Use `snake_case` for column names (e.g., `rider_count`, `stop_id`).

- Foreign key columns should explicitly reference the dimension (e.g., `stop_id`, `line_id`).

3. Primary Keys:

- Use `id` or `key` as suffixes for primary keys (e.g., `fact_id`, `time_key`).

4. Time Attributes:

- Use consistent naming like `datetime`, `hour`, `is_peak_hour`.

Project Standards

1. Version Control:

- Use Git for all project code and schema changes.

2. ETL Design:

- Modular ETL scripts using dbT.

- Each fact table has a dedicated ETL script.

3. Documentation:

- Maintain documentation for all dimension and fact table definitions.
- Use Power BI dashboards to visually track KPIs and testing results.

4. Testing:

- Validate data at each ETL stage.
- Regularly audit metrics like rider counts and delay times for accuracy.

4A AND 4B. TRANSIT DATA WAREHOUSE IMPLEMENTATION

1. SAMPLE SOURCE TO TARGET MAPPING

Staging Layer

Source	Target	Transformations
CSV.Fact_ID	RAW_TRANSIT_DATA.Fact_ID	Direct Load
CSV.Datetime	RAW_TRANSIT_DATA.Datetime	Convert to TIMESTAMP
CSV.Stop_ID	RAW_TRANSIT_DATA.Stop_ID	Direct Load

Dimension Tables

Source	Target	Transformations
stg_raw_transit.datetime,hour,day,month,year	DIM_TIME	Generate time_key, calculate is_weekend
stg_raw_transit.stop_*	DIM_STOPS	Direct mapping with deduplication
stg_raw_transit.line_*	DIM_LINES	Direct mapping with deduplication
stg_raw_transit.remote_unit_*	DIM_REMOTE_UNITS	Direct mapping with deduplication

Fact Tables

Source	Target	Transformations
stg_raw_transit	FACT_RIDER_VOLUME	Join with dimensions, calculate utilization

stg_raw_transit	FACT_STOP_UTILIZATION	Aggregate by stop, calculate metrics
stg_raw_transit	FACT_LINE_PERFORMANCE	Calculate performance metrics
stg_raw_transit	FACT_REMOTE_MONITOR	Track events and status
stg_raw_transit	FACT_RIDER_SEGMENTS	Segment analysis and categorization

2. LOADING SEQUENCE

1. Initial Load:

- Load CSV to RAW_TRANSIT_DATA
- Transform to staging view
- Load dimension tables
- Load fact tables

2. Incremental Updates:

- Append new data to RAW_TRANSIT_DATA
- Update staging view
- Update dimension tables (SCD Type 1)
- Append to fact tables

3. DATA QUALITY CHECKS

1. Staging Layer:

- No null keys

- Valid dates

- Valid numeric values

2. Dimension Tables:

- Unique keys

- No orphaned references

- Complete attributes

3. Fact Tables:

- Valid foreign keys

- Business rule validations

- Metric range checks

4. IMPLEMENTATION STANDARDS

1. Naming Conventions:

- Staging: stg_*

- Dimensions: dim_*

- Facts: fact_*

- Keys: *_id, *_key

2. Data Types:

- Keys: INTEGER

- Dates: TIMESTAMP

- Text: VARCHAR

- Metrics: FLOAT/INTEGER

3. Architecture:

- Three-layer architecture (STAGING, ODS, DWH)
- Star schema design
- Conformed dimensions

5. DATA PIPELINE (ETL/ELT):

5a. Code to load data into stage:

```
Create or replace TABLE TRANSIT_DWH.STAGING.RAW_TRANSIT_DATA (
    "Unnamed: 0" NUMBER(38,0),
    FACT_ID NUMBER(38,0),
    DATETIME TIMESTAMP_NTZ(9),
    STOP_ID NUMBER(38,0),
    REMOTE_UNIT_ID NUMBER(38,0),
    LINE_ID NUMBER(38,0),
    RIDER_COUNT NUMBER(38,0),
    STOP_NAME VARCHAR(255),
    NORTH_DIRECTION_LABEL VARCHAR(50),
    SOUTH_DIRECTION_LABEL VARCHAR(50),
    DIVISION VARCHAR(50),
    LINE VARCHAR(255),
    CONNECTING_LINES VARCHAR(255),
    DAYTIME_ROUTES VARCHAR(255),
    REMOTE_UNIT VARCHAR(50),
    HOUR NUMBER(38,0),
```

DAY VARCHAR(10),

MONTH NUMBER(38,0),

YEAR NUMBER(38,0),

CAPACITY NUMBER(38,0),

PEAK_OFF_PEAK VARCHAR(10),

SCHEDULE_ON_TIME VARCHAR(50),

TRIP_TIMING FLOAT,

EVENT_TYPE VARCHAR(50),

STATUS VARCHAR(50),

BATTERY_HEALTH FLOAT,

DEMOGRAPHIC VARCHAR(50),

TRAVEL_PATTERN VARCHAR(50),

ENTRIES NUMBER(38,0),

EXITS NUMBER(38,0),

NEAR_COMMERCIAL_AREA BOOLEAN,

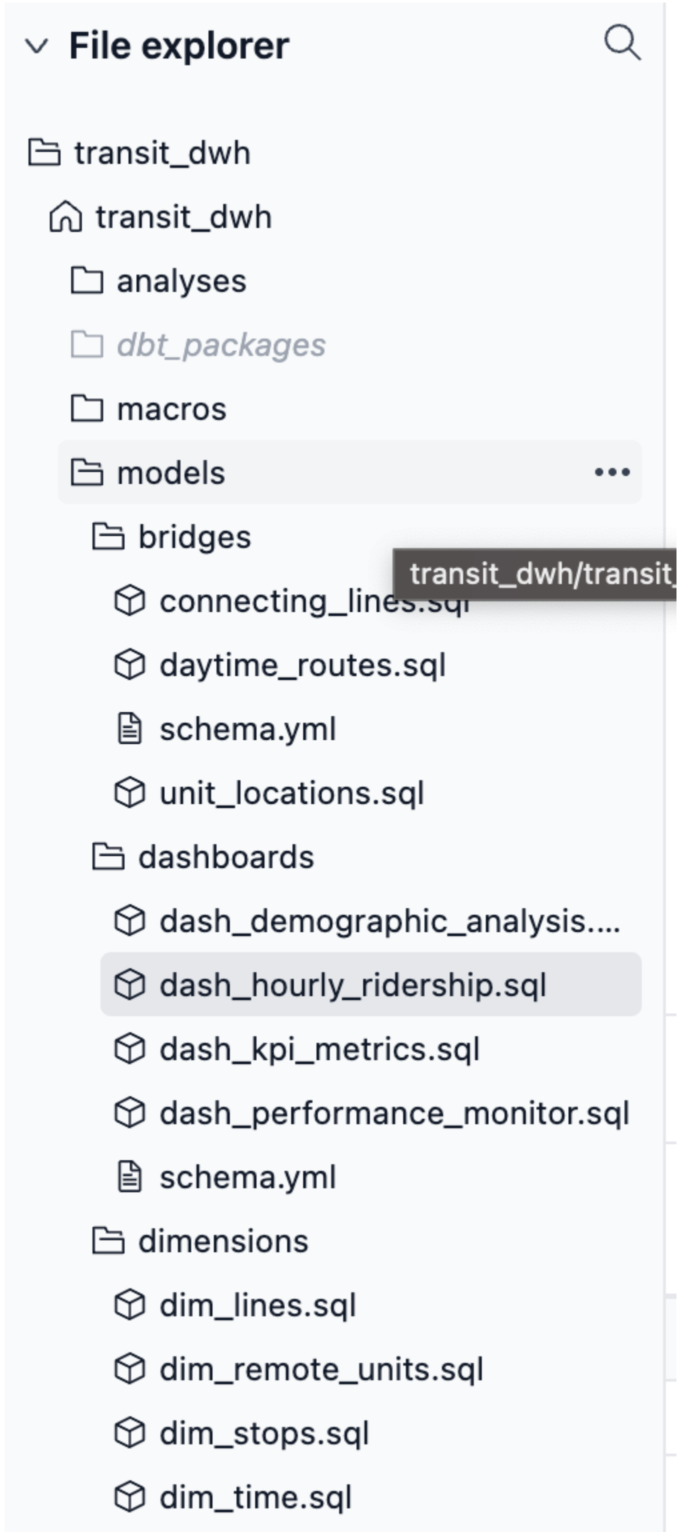
NEAR_UNIVERSITY BOOLEAN

);

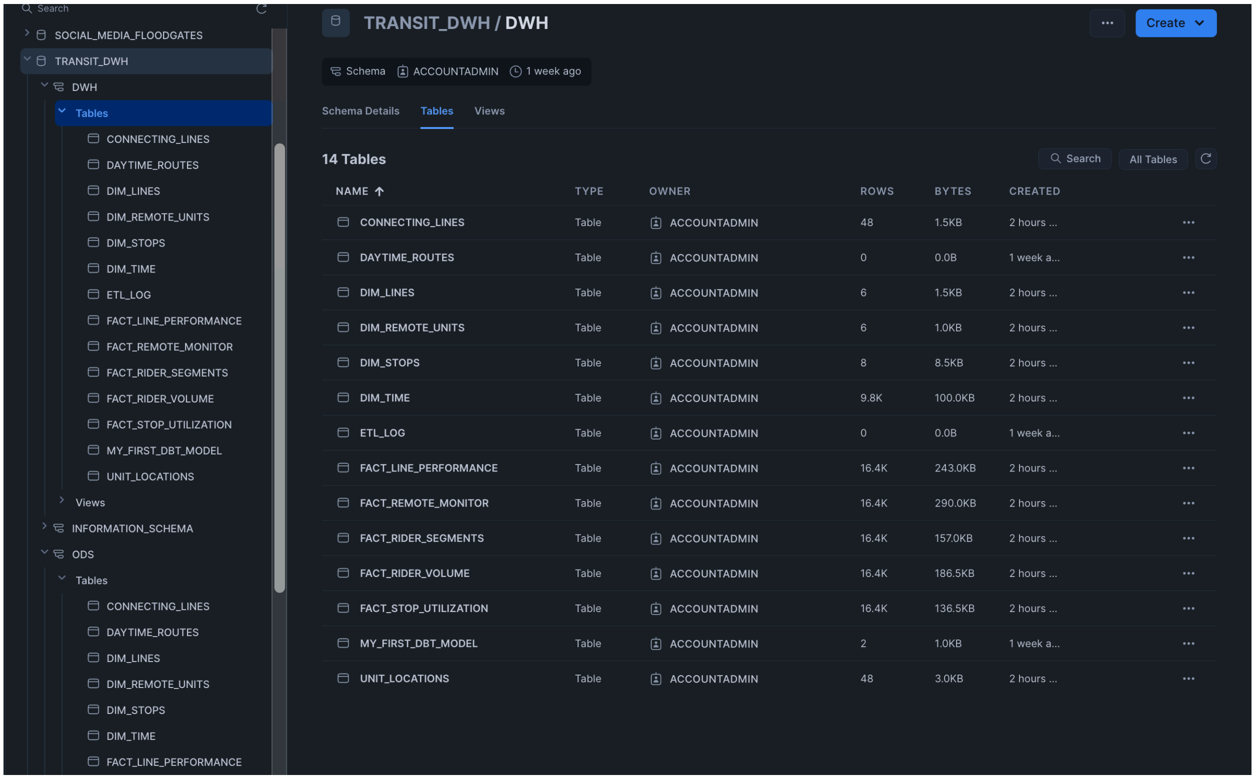
Data from our csv file was loaded through the data load wizard in snowflake.

5a. Data Lineage Chart

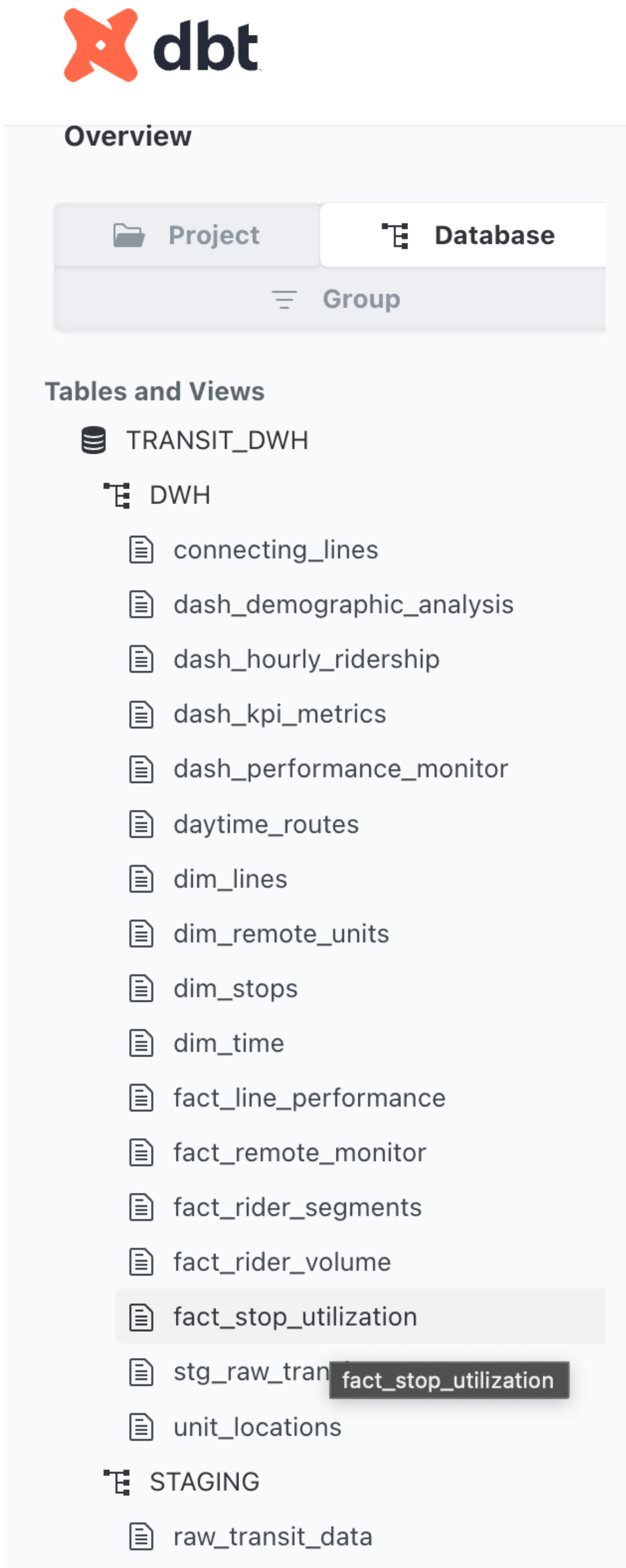
5b. Implementation from dBT file explorer


File explorer
transit_dwh
transit_dwh
analyses
dbt_packages
macros
models
bridges
transit_dwh/transit_
connecting_lines.sql
daytime_routes.sql
schema.yml
unit_locations.sql
dashboards
dash_demographic_analysis....
dash_hourly_ridership.sql
dash_kpi_metrics.sql
dash_performance_monitor.sql
schema.yml
dimensions
dim_lines.sql
dim_remote_units.sql
dim_stops.sql
dim_time.sql

5b. Snowflake Integration



5c. List of created tables in snowflake – generated via dbT documentation


dbt
Overview
Project
Database
Group
Tables and Views
TRANSIT_DWH
DWH
connecting_lines
dash_demographic_analysis
dash_hourly_ridership
dash_kpi_metrics
dash_performance_monitor
daytime_routes
dim_lines
dim_remote_units
dim_stops
dim_time
fact_line_performance
fact_remote_monitor
fact_rider_segments
fact_rider_volume
fact_stop_utilization
stg_raw_tran
fact_stop_utilization
unit_locations
STAGING
raw_transit_data

5d. Output from dbt build command to look for tests, data governance

stg_raw_transit	0.48s
dim_lines	1.78s
dim_remote_units	1.41s
dim_stops	1.76s
dim_time	1.49s
unit_locations	1.45s
not_null_dim_remote_units_remote_unit	0.22s
not_null_dim_remote_units_remote_unit_id	0.44s
unique_dim_remote_units_remote_unit_id	0.20s
not_null_unit_locations_location_id	0.24s
not_null_unit_locations_remote_unit_id	0.15s
not_null_unit_locations_stop_id	0.18s
relationships_unit_locations_remote_unit_id__remote_unit_id__ref_dim_remote_units_	0.24s
unique_unit_locations_location_id	0.15s
not_null_dim_time_datetime	0.20s
not_null_dim_time_time_key	0.16s
unique_dim_time_time_key	0.35s
not_null_dim_stops_stop_id	0.30s

6. BUSINESS INTELLIGENCE

We have created an initial version of a dashboard that displays various key metrics to gain insights from our business processes. Some of this include segmenting riders based on their age, or by the nature of the stops they enter or exit in (commercial, educational etc.). Additionally, we have also created key metrics to analyze battery health, delays in time, utilization rate etc. While this is an initial version of the dashboard, the final expected output would be to have a dashboard for each of the business processes.

We built the dashboard using Power BI after creating OBT Views in dBT. Our warehouse and database on Snowflake was connected to Power BI and then the dashboard was built.

7. REFLECTIONS AND CHALLENGES

Through this project, we were able to showcase our ability to identify business requirements and convert them to a functional reporting tool that is easily reproducible and updatable with new data. The project allowed us to delve deep and gain knowledge about Snowflake, dbT, Dimensional Modelling, and creating documentation for such a project.

However, we did face challenges with respect to data quality. One of the columns, which is a good to have but not a must have column, ended up having erroneous values, stopping us from creating a bridge table for it. The quality of data remains the utmost predictor of how good a datawarehouse can be. This ended up being our primary and only challenge that we faced in the whole project.

The next steps in this project would be to orchestrate inflow of new data using Apache Airflow, and to create multiple dashboards for each of the business processes, instead of an all-in-one dashboard. That would allow more enhanced insights on each of the business processes.