**SNCF Data modelling**

This Document will help you to understand the SNCF dataset, its ER diagram and its data modelling using Star Schema. At the End there are SQL queries for the given analysis in the assignment.

**SNCF Dataset:**

SNCF dataset is provided by GTFS which stands for General Transit Feed Specification, and it is a standardized format for public transportation schedules and geographic data. GTFS was developed by Google in collaboration with transit agencies to make it easier for developers to access and use public transit data. The GTFS dataset typically includes information about routes, stops, schedules, and geographic information related to public transportation systems, such as buses, trams, subways, and ferries.

There are total 10 files in the dataset and as per the assignment we are going to concentrate only on 5 files which are

* Trips.txt
* Stops.txt
* Stop\_times.txt
* Transfer.txt
* Routes.txt

**Trips.txt:**

Specific schedules for individual trips on the routes, including trip IDs, service IDs, and the sequence of stops for each trip.

There are total 7 attributes which are

1. Trips\_id: Identifies a trip.
2. Route\_id: ID referencing routes.route\_id. Identifies a route.
3. Service\_id: ID referencing services.service\_id. Identifies a service.
4. Trip\_headsign: Text that appears on signage identifying the trip's destination to riders. Use this field to distinguish between different patterns of service on the same route. If the headsign changes during a trip, trip\_headsign can be overridden by specifying values for the stop\_times.stop\_headsign.
5. Direction\_id: Indicates the direction of travel for a trip. This field is not used in routing; it provides a way to separate trips by direction when publishing timetables. Valid options are:
   1. 0 - Travel in one direction (e.g., outbound travel).
   2. 1 - Travel in the opposite direction (e.g., inbound travel).
6. Block\_id: Identifies the block to which the trip belongs. A block consists of a single trip or many sequential trips made using the same vehicle, defined by shared service days and block\_id. A block\_id can have trips with different service days, making distinct blocks.
7. Shape\_id: ID referencing shapes.shape\_id.Identifies a geospatial shape that describes the vehicle travel path for a trip.

**Stops.txt:** Description about the stop.

1. Stop\_id: Identifies the Stop.
2. Stop\_name: Name of the Stop.
3. Stop\_desc: Description about the Stop.
4. Stop\_lat: Latitude about the Stop.
5. Stop\_lon: Longitude about the Stop.
6. Zone\_id: Id refers to Zone.Zone\_id. Identifies the Zone.
7. Stop\_url: Url of the web page of the location.
8. Location\_type: Type of the location. Platform, Station, Entrance, Exit, Boarding Area or generic node.

• 0 (or empty): Stop (or Platform). A location where passengers board or disembark from a transit vehicle. Is called a platform when defined within a parent\_station.  
• 1: Station. A physical structure or area that contains one or more platform.  
• 2: Entrance/Exit. A location where passengers can enter or exit a station from the street. If an entrance/exit belongs to multiple stations, it can be linked by pathways to both, but the data provider must pick one of them as parent.  
• 3: Generic Node. A location within a station, not matching any other location\_type, which can be used to link together pathways define in pathways.txt.  
• 4: Boarding Area. A specific location on a platform, where passengers can board and/or alight vehicles.

1. Parent\_station: Defines hierarchy between the different locations defined in stops.txt. It contains the ID of the parent location.

• Stop/platform (location\_type=0): the parent\_station field contains the ID of a station.  
• Station (location\_type=1): this field must be empty.  
• Entrance/exit (location\_type=2) or generic node (location\_type=3): the parent\_station field contains the ID of a station (location\_type=1)  
• Boarding Area (location\_type=4): the parent\_station field contains ID of a platform.

**Stop\_times.txt**:

* trip\_id: Id refers to trips.trip\_id. Identifies a trip.
* arrival\_time: Arrival time at a specific stop for a specific trip on a route.
* departure\_time: Departure time at a specific stop for a specific trip on a route
* stop\_id: Id refers to stops.stop\_id. Identifies the stop.
* stop\_sequence: Order of stops for a particular trip. The values must increase along the trip but do not need to be consecutive.
* stop\_headsign: Text that appears on signage identifying the trip's destination to riders. This field overrides the default trips.trip\_headsign when the headsign changes between stops. If the headsign is displayed for an entire trip, use trips.trip\_headsign instead.
* pickup\_type: Indicates pickup method. Valid options are:
  + 0 or empty - Regularly scheduled pickup.
  + 1 - No pickup available.
  + 2 - Must phone agency to arrange pickup.
  + 3 - Must coordinate with driver to arrange pickup.
* drop\_off\_type: Indicates drop off method. Valid options are:  
   0 or empty - Regularly scheduled drop off.  
   1 - No drop off available.  
   2 - Must phone agency to arrange drop off.  
   3 - Must coordinate with driver to arrange drop off.
* shape\_dist\_traveled: Actual distance traveled along the associated shape, from the first stop to the stop specified in this record. This field specifies how much of the shape to draw between any two stops during a trip. Must be in the same units used in shapes.txt.

**routes.txt:**

1. route\_id: Identifies the route.
2. agency\_id: Id refers to Agency.agency\_id. Identifies the agency for the specified route..
3. route\_short\_name: Short name of the route.
4. route\_long\_name: long name of the route.
5. route\_desc: Description of a route that provides useful, quality information. Do not simply duplicate the name of the route.
6. route\_type: Indicates the type of transportation used on a route. Valid options are:  
   0 - Tram, Streetcar, Light rail. Any light rail or street level system within a metropolitan area.  
   1 - Subway, Metro. Any underground rail system within a metropolitan area.  
   2 - Rail. Used for intercity or long-distance travel.  
   3 - Bus. Used for short- and long-distance bus routes.  
   4 - Ferry. Used for short- and long-distance boat service.  
   5 - Cable tram. Used for street-level rail cars where the cable runs beneath the vehicle, e.g., cable car in San Francisco.  
   6 - Aerial lift, suspended cable car (e.g., gondola lift, aerial tramway). Cable transport where cabins, cars, gondolas or open chairs are suspended by means of one or more cables.  
   7 - Funicular. Any rail system designed for steep inclines.  
   11 - Trolleybus. Electric buses that draw power from overhead wires using poles.  
   12 - Monorail. Railway in which the track consists of a single rail or a beam.
7. route\_url: URL of a web page about the particular route. Should be different from the agency.agency\_url value.
8. route\_color: Route color designation that matches public facing material. Defaults to white (FFFFFF) when omitted or left empty.
9. route\_text\_color: Legible color to use for text drawn against a background of route\_color. Defaults to black (000000) when omitted or left empty.

**Transfer.txt**:

1. from\_stop\_id: Refers to stop.stop\_id. Identifies a stop or station where a connection between routes begins. If this field refers to a station, the transfer rule applies to all its child stops.
2. to\_stop\_id: Refers to stop.stop\_id. Identifies a stop or station where a connection between routes ends. If this field refers to a station, the transfer rule applies to all child stops.
3. transfer\_type: Indicates the type of connection for the specified (from\_stop\_id, to\_stop\_id) pair. Valid options are:
   1. 0 or empty - Recommended transfer point between routes.
   2. 1 - Timed transfer point between two routes. The departing vehicle is expected to wait for the arriving one and leave sufficient time for a rider to transfer between routes.
   3. 2 - Transfer requires a minimum amount of time between arrival and departure to ensure a connection. The time required to transfer is specified by min\_transfer\_time.
   4. 3 - Transfers aren't possible between routes at the location.
   5. 4 - Passengers can stay onboard the same vehicle to transfer from one trip to another (an "in-seat transfer").
   6. 5 - In-seat transfers aren't allowed between sequential trips. The passenger must alight from the vehicle and re-board.
4. min\_transfer\_time: Amount of time, in seconds, that must be available to permit a transfer between routes at the specified stops. The min\_transfer\_time should be sufficient to permit a typical rider to move between the two stops, including buffer time to allow for schedule variance on each route.
5. from\_route\_id: Refers to route.route\_id. Identifies a route where a connection between routes begins. If this field refers to a route.
6. to\_route\_id: Refers to route.route\_id. Identifies a route where a connection between routes begins. If this field refers to a route.

**Exercise:**

1. Build a Relational database model from the dataset

a. Add the data types to your tables.

b. Identify the PK, FK, UK for your tables.

c. Identify the relationship between tables & cardinality.

In the given draw.io file the first diagram refers to ER diagram of SNCF dataset.

Datatypes of the attribute, PK and FK are given below:

**Trips.txt:**

* attributes: trip\_id(varchar), route\_id(varchar), service\_id(int), trip\_headsign(text), direction\_id( enum), block\_id( int), shape\_id(int).

PK : trip\_id

FK: route\_id(route.route\_id), service\_id(service.service\_id), shape\_id(shape.shape\_id)

**Routes.txt:**

* Attributes: route\_id(varchar), agency\_id(varchar), route\_short\_name(text), route\_long\_name(text), route\_desc(text), route\_type(enum), route\_url(varchar), route\_color(varchar), route\_text\_color(varchar).
* PK: route\_id
* FK: agency\_id(agency.agency\_id)

**Stops.txt:**

* Attributes: stop\_id(varchar), stop\_name(text), stop\_desc(text), stop\_lat(float), stop\_lon(float), zone\_id(float), stop\_url(varchar), location\_type(enum), parent\_station(varchar).
* PK: Stop\_id

**Stop\_times.txt**:

* Attributes: trip\_id(varchar), arrival\_time(time), departure\_time(time), stop\_id(varchar), stop\_sequence(int), stop\_headsign(text), pickup\_type(enum), drop\_off\_type(enum), shape\_dist\_traveled(float).
* FK: trip\_id(trip.trip\_id), stop\_id(stops.stop\_id)

**Transfer.txt:**

* Attributes: from\_stop\_id(varchar), to\_stop\_id(varchar), transfer\_type(int), min\_transfer\_time(int), from\_route\_id(varchar), to\_route\_id(varchar).
* FK: from\_stop\_id, to\_stop\_id(stops.stop\_id), from\_route\_id, to\_route\_id(route.route\_id)

ER diagram can be found in draw.io file under data\_modelling folder. Below is my assumption taken while creating the relationship and cardinality.

Assumption made while designing the ER diagram:

1. Route is in relation with trips table by route\_id. One route can have many trips but 1 trip can have only one route. So, the relationship between them is one to many. And also, route is related with transfer and one route may or may not have a transfer or have many transfers. but one transfer should have one or many routes. So, the relationship between them is many optional to many mandatory.
2. Stop is in relation to stop\_times. One stop should have one or many stop times. And one stop time should have one stop. For ex Stop A has 3 times (6 AM, 7 AM, 9 AM) but at 6 AM for that stop is one. So, the relationship between is one mandatory to many mandatory. Stop is also in relation to transfer. One stop may have one, many or no transfer. But one transfer should have one stop only. So, the relationship between them Is many optional to one mandatory.
3. Trip is in relation to stop\_times. One trip can have many stops time, but one stop time will have one trip. So, the relationship between is one mandatory to many mandatory.
4. And Also in stop column we have parent\_station attribute which a self-referencing to Stop table itself.

2. **Convert your Relational database model to Star Schema model**

a. **Bonus**: Add technical columns, Index, Partition, Cluster columns.

The second diagram in draw.io file is a conversion of ER diagram to Star Schema.

Steps I took to convert ER diagram to Star Schema:

1. Star Schema refers to the design of tables in such that, the fact table is surrounded by dimension tables in a star pattern.
2. Fact tables record measurements or metrics for a specific event. And dimension tables give the context to the respective attributes of the fact table.
3. As per the Kimball, the fact tables are selected based on the lowest granularity and the attributes represent the measurements.
4. Based on that, I have chosen to combine stop\_times and transfer tables as my fact table because it is the lowest grain, By stop\_times and tranfer u can identify the trip, route, and stop. And there are attributes which are quantitative.
5. Moreover, both tables had FK and If I take either of the table as dim table then it would have relation with another dim Table which will cause circular relation.
6. In the Fact table I have added a new attribute called Date which can be taken from trip\_id itself.
7. Note: I have removed many attributes from the tables, just because I have to keep the tables and attributes related to this assignment. Like service\_id, agency\_id was not relevant to this assignment. If I had to use the service and agency table, I would have considered using those attributes as well.
8. From the fact table we can measure the time taken between each stop, most pickup\_time used in the route.
9. Note: The fact table is also selected based on the business case. What you want to analyze.
10. And the 3 queries provided in assignment can be easily queried by the star schema proposed.

**Indexing and Partitioning:**

* As we have date column in fact table which is extracted from trip\_id. We can do partitioning based on Year, Month or day as I have done in Python Exercise.
* And Indexing I chose to do with stop\_id and route\_id because indexing are done on columns which are not updated frequently orelse it would lead to slower insert/update operations. As per my assumption stop\_id and route\_is not updated regularly and trip\_id can be dynamic (Suppose, for Each traversal from Stop A to Stop B there might be different id). And moreover, indexing are done on the column which is most queried. As given in you exercise too.

**3. SQL Exercise - Based on your Star Schema model, provide sample queries for data analysts**

**a. Top 10 most popular routes in August 2022**

- SELECT count(route\_id) AS count\_of\_routes\_used, route\_id AS popular\_routes FROM fct\_stop\_times st JOIN dim\_trips t on st.trip\_id=t.trip\_id WHERE date BETWEEN 01/08/2022 AND 30/08/2022 GROUP BY popular\_routes ORDER BY popular\_routes LIMIT 10

**b. Routes with number of stops in descending order.**

- SELECT distinct route\_id AS route, COUNT (stop\_id) as num\_of\_stops FROM fct\_stop\_times GROUP BY route\_id ORDER BY num\_of\_stops DESC

**c. Number of trips with missed transfers in August 2022**

- SELECT COUNT (trip\_id) AS num\_of\_trips from fct\_stop\_times st join trip t on st.trip\_id = t.trip\_id WHERE date BETWEEN 01/08/2022 AND 30/08/2022 AND st.to\_stop\_id=NULL.