

DATA WAREHOUSING END SEM TASK

GOM4DW to Snowflake schema

Group 16:

Rahul Maheshwari (MT19027)

P. Akshay Kumar (MT19094)

Nikunj Agarwal (MT19093)

Objective

To add functionality to the existing tool of converting a GOM4DW schema to Snowflake schema and run it on Vyapari Dataset. This tool will have a button 'Convert to ROLAP' that will perform this activity.

To run this tool, please execute **WelcomeScreen.java** if you want to work on a new project. If you want to launch an already existing project then execute **LoginScreen.java** and provide project name as **P1**. The java files are located in **DW_endSemProj\src\com\FinalInfo**

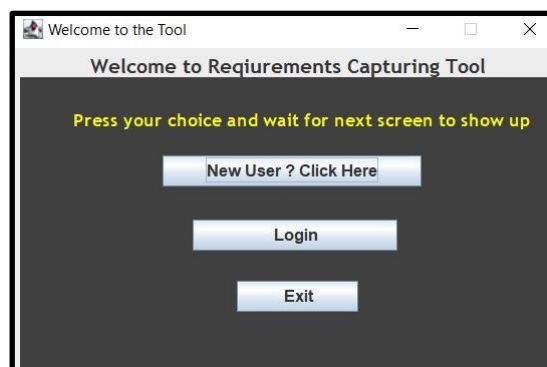
The database P1 is exported to an excel file named **P1.xlsx** and this is included in the project folder.

To import this, excel file as database, please refer to the following link.

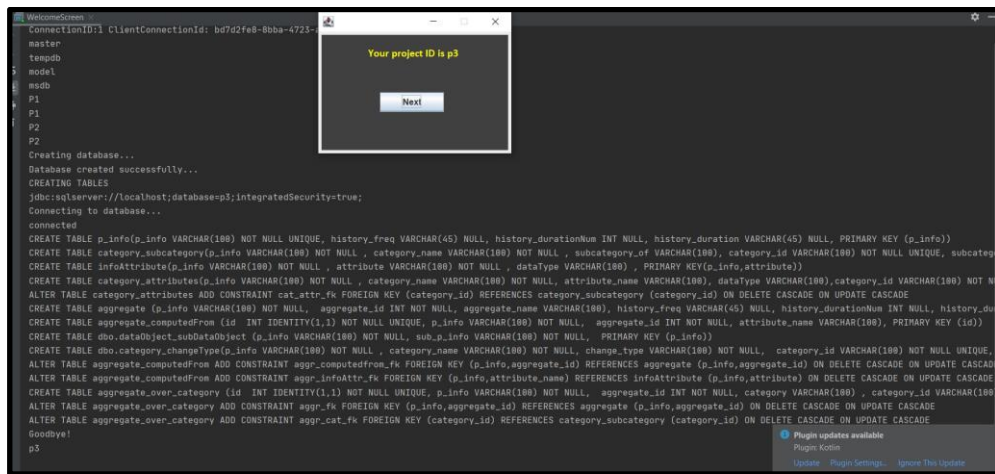
<https://www.sqlserverlogexplorer.com/import-and-export-database/>

How was the tool used to capture the GOM4DW data and category objects?

Step 1: When the welcome screen was launched, the following GUI showed up.



Step 2: Once we click on “New User? Click Here”, the tool generates all the tables required to capture the data and category objects.

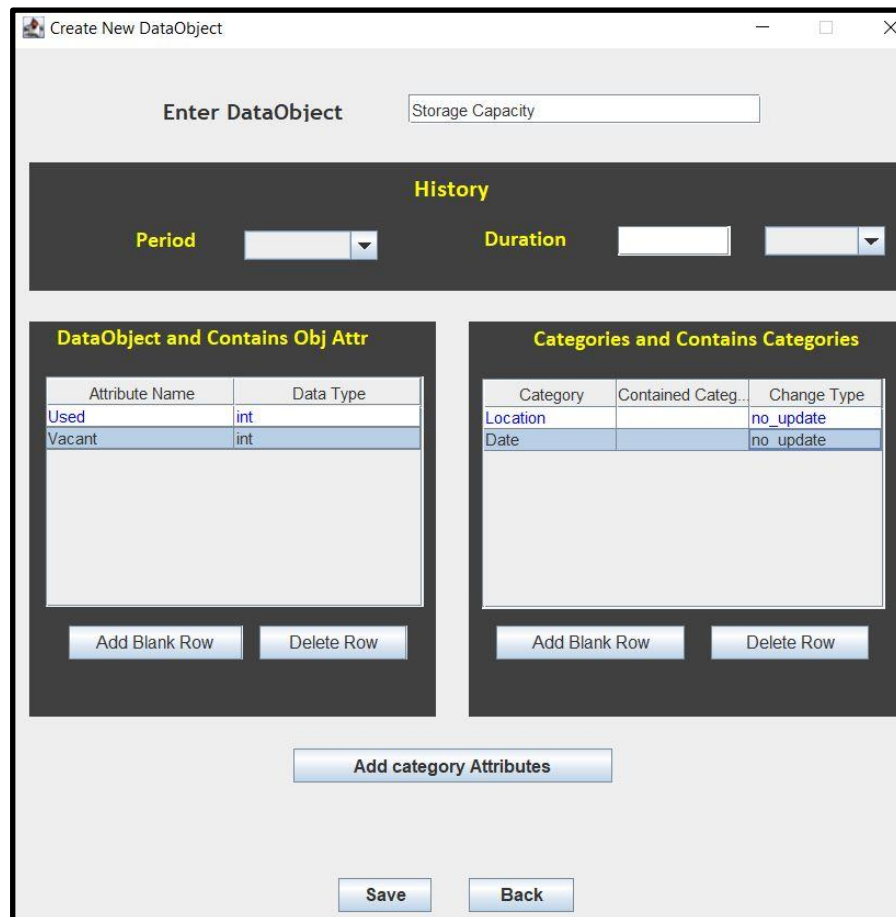


The screenshot shows a terminal window with the following content:

```
ConnectionID: bd7d2fe8-8bba-4723-
master
tempdb
model
msdb
P1
P2
P2
Creating database...
Database created successfully...
CREATING TABLES
jdbc:sqlserver://localhost:database=p3;integratedSecurity=true;
Connecting to database...
connected
CREATE TABLE p_info(p_info VARCHAR(100) NOT NULL UNIQUE, history_freq VARCHAR(45) NULL, history_durationNum INT NULL, history_duration VARCHAR(45) NULL, PRIMARY KEY (p_info))
CREATE TABLE category_subcategory(p_info VARCHAR(100) NOT NULL, category_name VARCHAR(100) NOT NULL, subcategory_of VARCHAR(100), category_id VARCHAR(100) NOT NULL UNIQUE, subcategory_id VARCHAR(100) NOT NULL, attribute_name VARCHAR(100), dataType VARCHAR(100), PRIMARY KEY (p_info, attribute_name))
CREATE TABLE category_attributes(p_info VARCHAR(100) NOT NULL, category_name VARCHAR(100) NOT NULL, attribute_name VARCHAR(100), dataType VARCHAR(100), category_id VARCHAR(100) NOT NULL, subcategory_id VARCHAR(100) NOT NULL, attribute_value VARCHAR(100), PRIMARY KEY (p_info, attribute_name))
ALTER TABLE category_attributes ADD CONSTRAINT cat_attr_fk FOREIGN KEY (category_id) REFERENCES category_subcategory (category_id) ON DELETE CASCADE ON UPDATE CASCADE
CREATE TABLE aggregate(p_info VARCHAR(100) NOT NULL, aggregate_id INT NOT NULL, aggregate_name VARCHAR(100), history_freq VARCHAR(45) NULL, history_durationNum INT NULL, history_duration VARCHAR(45) NULL, PRIMARY KEY (p_info, aggregate_id))
CREATE TABLE aggregate_computedFrom (id INT IDENTITY(1,1) NOT NULL UNIQUE, p_info VARCHAR(100) NOT NULL, aggregate_id INT NOT NULL, attribute_name VARCHAR(100), PRIMARY KEY (id))
CREATE TABLE dbo.dataObject_subDataObject (p_info VARCHAR(100) NOT NULL, sub_p_info VARCHAR(100) NOT NULL, PRIMARY KEY (p_info))
CREATE TABLE dbo.category_changeType(p_info VARCHAR(100) NOT NULL, category_name VARCHAR(100) NOT NULL, change_type VARCHAR(100) NOT NULL, category_id VARCHAR(100) NOT NULL UNIQUE, PRIMARY KEY (p_info, category_name, change_type))
ALTER TABLE aggregate_computedFrom ADD CONSTRAINT aggr_computedfrom_fk FOREIGN KEY (p_info, aggregate_id) REFERENCES aggregate (p_info, aggregate_id) ON DELETE CASCADE ON UPDATE CASCADE
ALTER TABLE aggregate_computedFrom ADD CONSTRAINT aggr_attr_fk FOREIGN KEY (p_info, attribute_name) REFERENCES category_attributes (p_info, attribute_name) ON DELETE CASCADE ON UPDATE CASCADE
CREATE TABLE aggregate_over_category (id INT IDENTITY(1,1) NOT NULL UNIQUE, p_info VARCHAR(100) NOT NULL, aggregate_id INT NOT NULL, category VARCHAR(100), category_id VARCHAR(100), PRIMARY KEY (id))
ALTER TABLE aggregate_over_category ADD CONSTRAINT aggr_cat_fk FOREIGN KEY (p_info, aggregate_id) REFERENCES aggregate (p_info, aggregate_id) ON DELETE CASCADE ON UPDATE CASCADE
ALTER TABLE aggregate_over_category ADD CONSTRAINT aggr_cat_fk FOREIGN KEY (category_id) REFERENCES category_subcategory (category_id) ON DELETE CASCADE ON UPDATE CASCADE
Goodbye!
p3
```

A small dialog box titled "Your project ID is p3" with a "Next" button is overlaid on the terminal window.

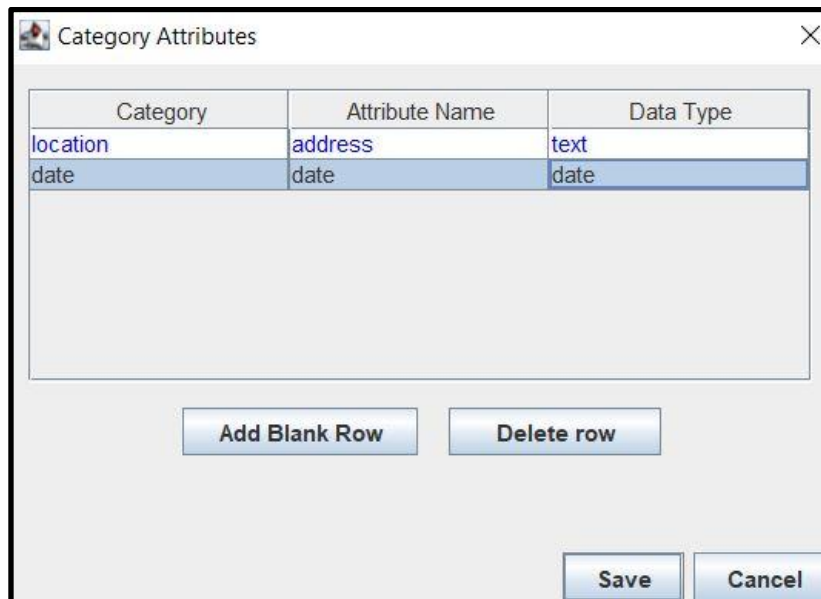
Step 3: Next. click on “Create new object” and make entries for all data objects, their attributes, categories, sub-categories (if any) and their change types.



The "Create New DataObject" dialog box contains the following elements:

- Enter DataObject:** A text input field containing "Storage Capacity".
- History:** A section with "Period" (a dropdown menu) and "Duration" (a text input field).
- DataObject and Contains Obj Attr:** A table with two columns: "Attribute Name" and "Data Type". It contains two rows: "Used" with "int" and "Vacant" with "int". Below the table are "Add Blank Row" and "Delete Row" buttons.
- Categories and Contains Categories:** A table with three columns: "Category", "Contained Categ...", and "Change Type". It contains two rows: "Location" with "no_update" and "Date" with "no_update". Below the table are "Add Blank Row" and "Delete Row" buttons.
- Add category Attributes:** A button located below the categories table.
- Save and Back:** Two buttons at the bottom of the dialog.

Step 4: We then click on add category attributes to add the attributes corresponding to each category.

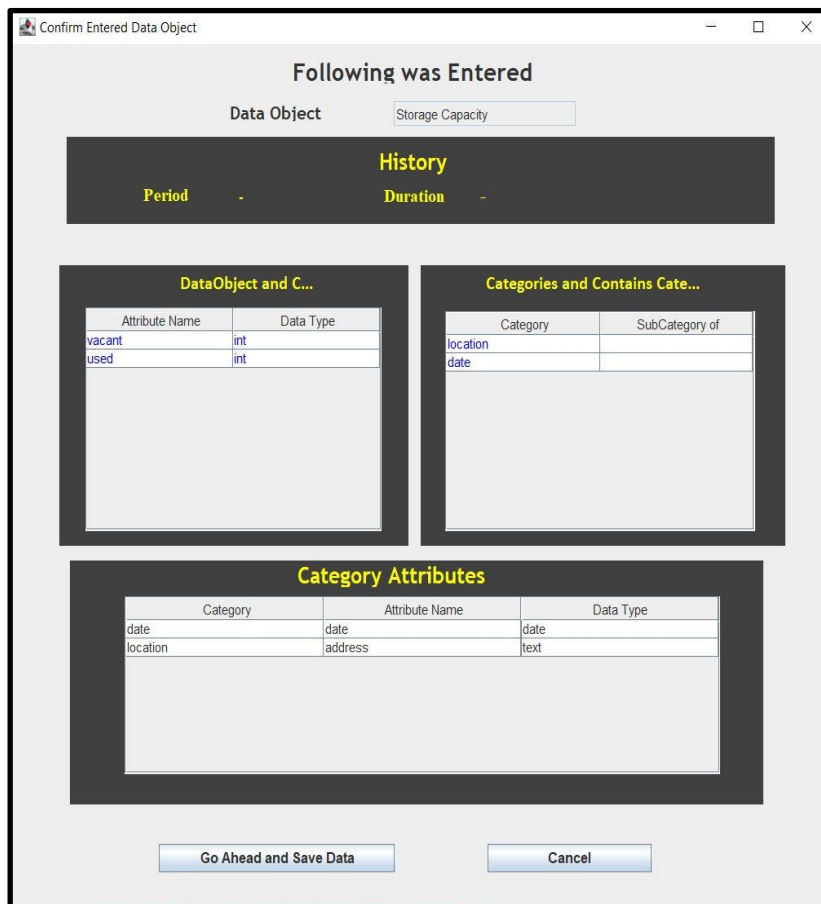


A dialog box titled "Category Attributes" with a close button (X) in the top right corner. It contains a table with three columns: "Category", "Attribute Name", and "Data Type". The table has two rows: the first row has "location" in the "Category" column, "address" in the "Attribute Name" column, and "text" in the "Data Type" column; the second row has "date" in the "Category" column, "date" in the "Attribute Name" column, and "date" in the "Data Type" column. Below the table are two buttons: "Add Blank Row" and "Delete row". At the bottom right are two buttons: "Save" and "Cancel".

Category	Attribute Name	Data Type
location	address	text
date	date	date

Buttons: Add Blank Row, Delete row, Save, Cancel

Step 5: Once we click on “Save” button, the GUI prompts for confirmation. On successful insertion, the GUI show success message “Saved successfully”.



A dialog box titled "Confirm Entered Data Object" with standard window controls (minimize, maximize, close) in the top right corner. It displays a summary of entered data. At the top, it says "Following was Entered". Below this, there is a "Data Object" section with a "Storage Capacity" input field. A "History" section shows "Period" and "Duration" both with a "-" value. There are two side-by-side sections: "DataObject and C..." and "Categories and Contains Cate...". The "DataObject and C..." section contains a table with two columns: "Attribute Name" and "Data Type", with rows for "vacant" (int) and "used" (int). The "Categories and Contains Cate..." section contains a table with two columns: "Category" and "SubCategory of", with rows for "location" and "date". At the bottom, there is a "Category Attributes" section with a table with three columns: "Category", "Attribute Name", and "Data Type", with rows for "date" (date), "location" (address), and "date" (text). At the bottom of the dialog are two buttons: "Go Ahead and Save Data" and "Cancel".

Following was Entered

Data Object: Storage Capacity

History: Period -, Duration -

DataObject and C...:

Attribute Name	Data Type
vacant	int
used	int

Categories and Contains Cate...:

Category	SubCategory of
location	
date	

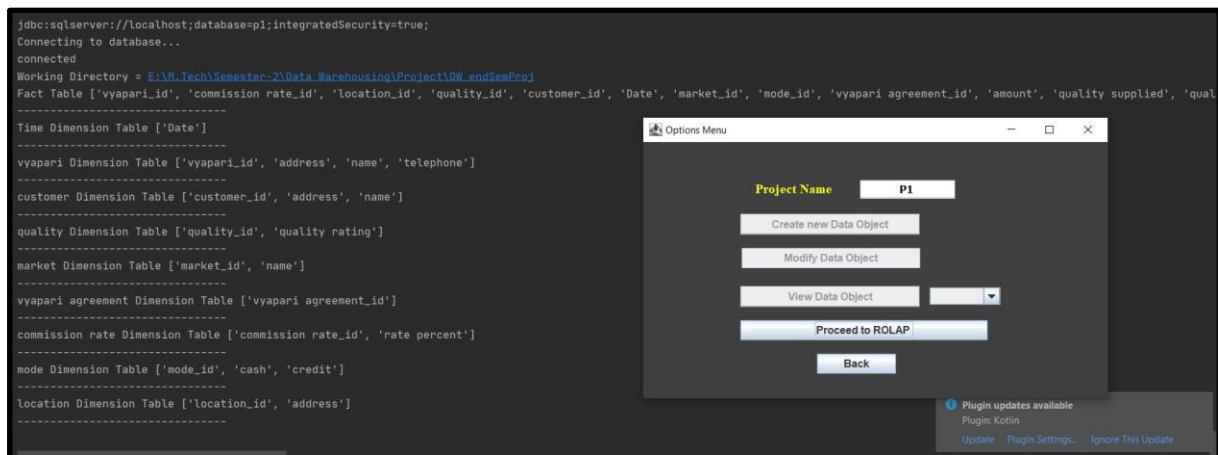
Category Attributes:

Category	Attribute Name	Data Type
date	date	date
location	address	text

Buttons: Go Ahead and Save Data, Cancel



Step 6: Proceed to ROLAP triggers button on the project home screen triggers the python script and ROLAP schema is generated as output.



How Data object within a data object was handled?

A datatype “Data Object” was added in the list of datatypes associated with attributes of a data object. This differentiates a data object from other attributes and this condition was used to check if another data object was contained in a data object.

How change type was handled?

Change type was added as a field in “Create User Info” screen and its corresponding table was created with the schemas as follows.

```
dbo.category_changeType(p_info varchar, category_name varchar, change_type varchar,
category_id varchar, time_stamp datetime, primary_key(category_id));
```

“change_type” was a combo list with values “no_change”, “type-1”, “type-2”. The default value of “time_stamp” was set to be CURRENT_TIMESTAMP and that of “change_type” was “no_change”

An entry in the change_type table is shown below.

	p_info	category_name	change_type	category_id	time_stamp
1	Agreed Commission	day	no_change	Agreed Commission_day	2020-05-18 21:41:08.557

Implementation of Conversion algorithm.

Database connection

Conversion algorithm was implemented in Python and the script gets triggered on click of “Proceed to OLAP” button on the “createNewInfo” screen on the GUI. Pyodbc object was used to connect to the MSSQL database.

Handling Data Objects

Each data object was stored in a ‘fact’ list and its associated attributes were stored in “facts_attributes_dict” with key as the ‘data object’ (fact). If the data object had other data objects as attributes, then the data object was added as a dimension along with its attributes to “dimensions_attributes_dict”. For every dimension, an attribute with name “dimension_id” was added to the dimension attributes list which will act as the primary key for dimension table when the ROLAP schema is built. This is also added to “foreign_key_list” which will be later added to the fact table since this will act as the foreign key constraint link between the fact table and the dimension table.

Handling category objects

Each category associated with the data object that is not present in the list of dimensions, is added to the dimensions list and its associated attributes are linked with the dimension. Also, if the change type is “no_change” and timestamp is NULL, then the timestamp is added as an attribute to the dimension dict. Each dimension is then linked to the data object which is initially fetched from database. If there are subcategories then a subcategory dictionary is created with subcategory as key and its attributes as values. This is then later linked with the associated dimension.

Handling the time attributes

All the date/day attributes associated with the “Potato Adhati vyapari” usecase are replaced with a single “Time Dimension” with attribute as (date,day,week,month,quarter,year) since most of the ROLAP operations that occurs with respect to this use case is operated on daily basis to retrieve useful information and maintaining history information in the system.

Output:

Facts:

['market_id', 'mode_id', 'vyapari_agreement_id', 'vyapari_id', 'quality_id', 'time_id', 'commission_rate_id', 'location_id', 'customer_id', 'quality_supplied', 'sgst', 'commission_amount', 'price', 'total_quantity_in_hand', 'quality', 'quantity', 'commission_rate_agreed', 'easy_quality_change', 'amount', 'easy_new_price', 'discount', 'quantity_supplied', 'punctual_supply', 'selling_rate', 'delivered_quality', 'commission_rate', 'price_agreed', 'original_price_agreed', 'quality_agreed', 'used', 'vacant', 'price_in_market', 'delivery_time_price_agreed', 'delivery_lead_time', 'quantity_agreed', 'quantity_sold_in_market', 'quality_required', 'transport_cost', 'quantity_thrown', 'agreed_commission_rate', 'easy_new_quantity', 'storage_cost', 'cgst', 'quantity_in_market', 'quantity_required']

Dimensions:

Time: ['time_id', 'date', 'day', 'week', 'month', 'quarter', 'year']

vyapari: ['vyapari_id', 'name', 'telephone', 'address']

customer: ['customer_id', 'name', 'address']

quality: ['quality_id', 'quality_rating']

market: ['market_id', 'name']

vyapari agreement: ['vyapari_agreement_id']

commission rate: ['commission_rate_id', 'rate_percent']

mode: ['mode_id', 'cash', 'credit']

location: ['location_id', 'address']

Create SQLs for FACT and DIMENSION tables:

```
CREATE TABLE DBO.vyapari_TABLE (vyapari_id int PRIMARY KEY,telephone int,name text,address text);
```

```
CREATE TABLE DBO.customer_TABLE (customer_id int PRIMARY KEY,name text,address text);
```

```
CREATE TABLE DBO.quality_TABLE (quality_id int PRIMARY KEY,quality_rating int);
```

```
CREATE TABLE DBO.market_TABLE (market_id int PRIMARY KEY,name text);
```

```
CREATE TABLE DBO.vyapari_agreement_TABLE (vyapari_agreement_id int PRIMARY KEY);
```

```
CREATE TABLE DBO.commission_rate_TABLE (commission_rate_id int PRIMARY KEY,rate_percent float(4));
```

```
CREATE TABLE DBO.mode_TABLE (mode_id int PRIMARY KEY,cash bit,credit bit);
```

```
CREATE TABLE DBO.location_TABLE (location_id int PRIMARY KEY,address text);
```

```
CREATE TABLE DBO.TIME_TABLE (time_id int PRIMARY KEY,date date,day int,week  
int,month int,quarter int,year int);
```

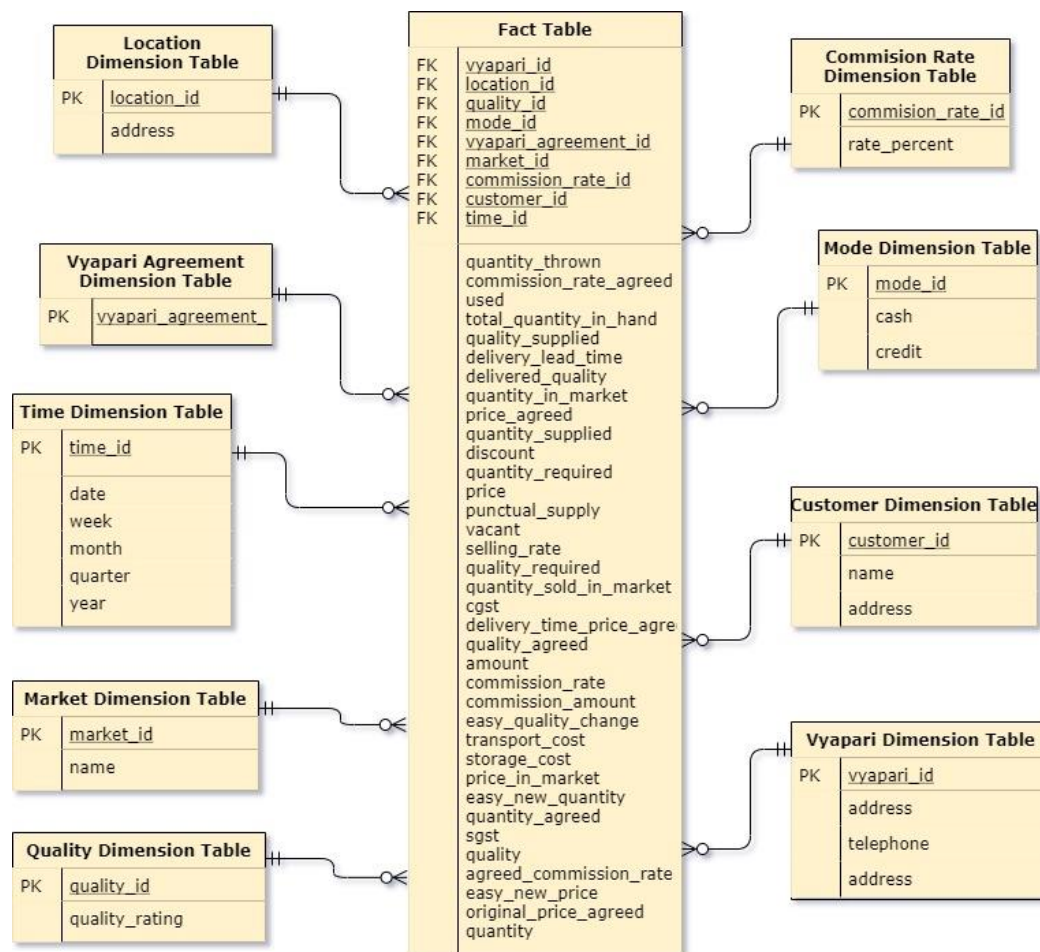
```
CREATE TABLE DBO.FACT_TABLE (market_id int REFERENCES  
market_TABLE(market_id),mode_id int REFERENCES  
mode_TABLE(mode_id),vyapari_agreement_id int REFERENCES  
vyapari_agreement_TABLE(vyapari_agreement_id),vyapari_id int REFERENCES  
vyapari_TABLE(vyapari_id),quality_id int REFERENCES  
quality_TABLE(quality_id),time_id int REFERENCES  
TIME_TABLE(time_id),commission_rate_id int REFERENCES  
commission_rate_TABLE(commission_rate_id),location_id int REFERENCES  
location_TABLE(location_id),customer_id int REFERENCES  
customer_TABLE(customer_id),quality_supplied text,sgst float(4),commission_amount  
float(4),price float(4),total_quantity_in_hand int,quality text,quantity  
int,commission_rate_agreed float(4),easy_quality_change int,amount  
float(4),easy_new_price int,discount float(4),quantity_supplied int,punctual_supply  
bit,selling_rate float(4),delivered_quality text,commission_rate float(4),price_agreed  
float(4),original_price_agreed float(4),quality_agreed text,used float(4),vacant  
float(4),price_in_market float(4),delivery_time_price_agreed float(4),delivery_lead_time  
float(4),quantity_agreed int,quantity_sold_in_market int,quality_required text,transport_cost  
float(4),quantity_thrown int,agreed_commission_rate float(4),easy_new_quantity  
int,storage_cost float(4),cgst float(4),quantity_in_market int,quantity_required int);
```



output.txt

* Refer to the output.txt file for detailed output

Following is the snowflake schema developed by code:



Screenshot for tables created in SQL Server:

SQLQuery1.sql - D:\CAMJHE\aniku (53))*

```
use p1;
select * from INFORMATION_SCHEMA.tables where TABLE_NAME like '%_TABLE';
```

100 %

Results Messages

	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE
1	p1	dbo	vyapari_TABLE	BASE TABLE
2	p1	dbo	customer_TABLE	BASE TABLE
3	p1	dbo	quality_TABLE	BASE TABLE
4	p1	dbo	market_TABLE	BASE TABLE
5	p1	dbo	vyapari_agreement_TABLE	BASE TABLE
6	p1	dbo	commission_rate_TABLE	BASE TABLE
7	p1	dbo	mode_TABLE	BASE TABLE
8	p1	dbo	location_TABLE	BASE TABLE
9	p1	dbo	TIME_TABLE	BASE TABLE
10	p1	dbo	FACT_TABLE	BASE TABLE

ASSUMPTIONS:

- “Change_type” can be set only at the time of creation of category in GUI and its addition to the database.
- “Data Object” is used as a datatype to identify data objects within a data object.
- If history is to be maintained for any data object, then the attributes '**day**', '**week**', '**month**', '**quarter**', '**year**' automatically get added to time dimension in order to support ROLAP operations.
- All date and day related attributes are assumed as date and added to the time dimension.