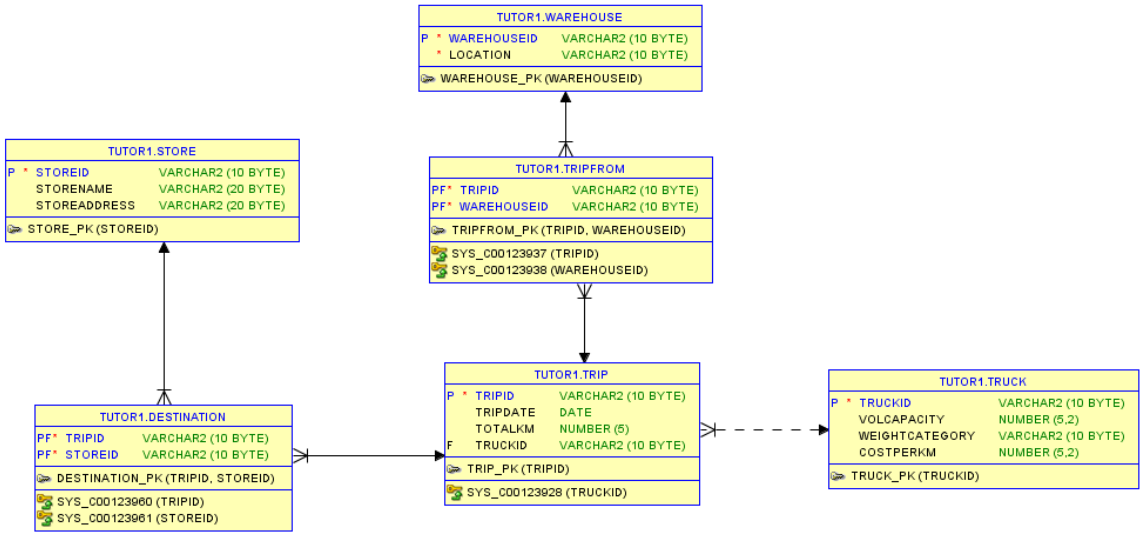
Laboratory 3
TRUCK DELIVERY CASE STUDY

1. A Truck Delivery Case Study – Description

A trucking company is responsible for picking up shipments from warehouses of a retail chain called MYER, and delivering the shipments to individual retail store of MYER. A truck may carry several shipments during a single trip, which is identified by TripID, and delivers those shipments to multiple stores. Trucks have different capacities for both the volumes they can hold and the weights they can carry. At the moment, a truck makes several trips each week. An operational database is being used to keep track the deliveries, including the scheduling of trucks, which provide timely deliveries to stores. The following is an E/R diagram of the truck delivery system.



The tables from the operational database are as follows:

Warehouse (**WarehouseID**, Location)
Trip (**TripID**, Date, TotalKm, *TruckID*)
TripFrom (**TripID, WarehouseID**)
Truck (**TruckID**, VolCapacity, WeightCategory, CostPerKm)
Store (**StoreID**, StoreName, Address)
Destination (**TripID, StoreID**)

First you need to create the operational database using the following SQL:

```
Create Table Warehouse
(WarehouseID  Varchar2(10) Not Null,
 Location     Varchar2(10) Not Null,
 Primary Key (WarehouseID)
);

Create Table Truck
(TruckID        Varchar2(10) Not Null,
 VolCapacity    Number(5,2),
 WeightCategory Varchar2(10),
```

```
 CostPerKm      Number(5,2),
 Primary Key (TruckID)
);

Create Table Trip
(TripID   Varchar2(10) Not Null,
 TripDate Date,
 TotalKm  Number(5),
 TruckID  Varchar2(10),
 Primary Key (TripID),
 Foreign Key (TruckID) References Truck(TruckID)
);

Create Table TripFrom
(TripID      Varchar2(10) Not Null,
 WarehouseID Varchar2(10) Not Null,
 Primary Key (TripID, WarehouseID),
 Foreign Key (TripID) References Trip(TripID),
 Foreign Key (WarehouseID) References Warehouse(WarehouseID)
);

Create Table Store
(StoreID      Varchar2(10) Not Null,
 StoreName    Varchar2(20),
 StoreAddress Varchar2(20),
 Primary Key (StoreID)
);

Create Table Destination
(TripID       Varchar2(10) Not Null,
 StoreID      Varchar2(10) Not Null,
 Primary Key (TripID, StoreID),
 Foreign Key (TripID) References Trip(TripID),
 Foreign Key (StoreID) References Store(StoreID)
);

--Insert Records to Operational Database
Insert Into Warehouse Values ('W1','Warehouse1');
Insert Into Warehouse Values ('W2','Warehouse2');
Insert Into Warehouse Values ('W3','Warehouse3');
Insert Into Warehouse Values ('W4','Warehouse4');
Insert Into Warehouse Values ('W5','Warehouse5');

Insert Into Truck Values ('Truck1', 250, 'Medium', 1.2);
Insert Into Truck Values ('Truck2', 300, 'Medium', 1.5);
Insert Into Truck Values ('Truck3', 100, 'Small',  0.8);
Insert Into Truck Values ('Truck4', 550, 'Large',  2.3);
Insert Into Truck Values ('Truck5', 650, 'Large',  2.5);

Insert Into Trip Values ('Trip1', to_date('14-Apr-2013', 'DD-MON-YYYY'), 370, 'Truck1');
Insert Into Trip Values ('Trip2', to_date('14-Apr-2013', 'DD-MON-YYYY'), 570, 'Truck2');
Insert Into Trip Values ('Trip3', to_date('14-Apr-2013', 'DD-MON-YYYY'), 250, 'Truck3');
Insert Into Trip Values ('Trip4', to_date('15-Jul-2013', 'DD-MON-YYYY'), 450, 'Truck1');
Insert Into Trip Values ('Trip5', to_date('15-Jul-2013', 'DD-MON-YYYY'), 175, 'Truck2');

Insert Into TripFrom Values ('Trip1', 'W1');
Insert Into TripFrom Values ('Trip1', 'W4');
Insert Into TripFrom Values ('Trip1', 'W5');
Insert Into TripFrom Values ('Trip2', 'W1');
Insert Into TripFrom Values ('Trip2', 'W2');
Insert Into TripFrom Values ('Trip3', 'W1');
Insert Into TripFrom Values ('Trip3', 'W5');
Insert Into TripFrom Values ('Trip4', 'W1');
Insert Into TripFrom Values ('Trip5', 'W4');
Insert Into TripFrom Values ('Trip5', 'W5');

Insert Into Store Values ('M1', 'Myer City', 'Melbourne');
Insert Into Store Values ('M2', 'Myer Chaddy', 'Chadstone');
```

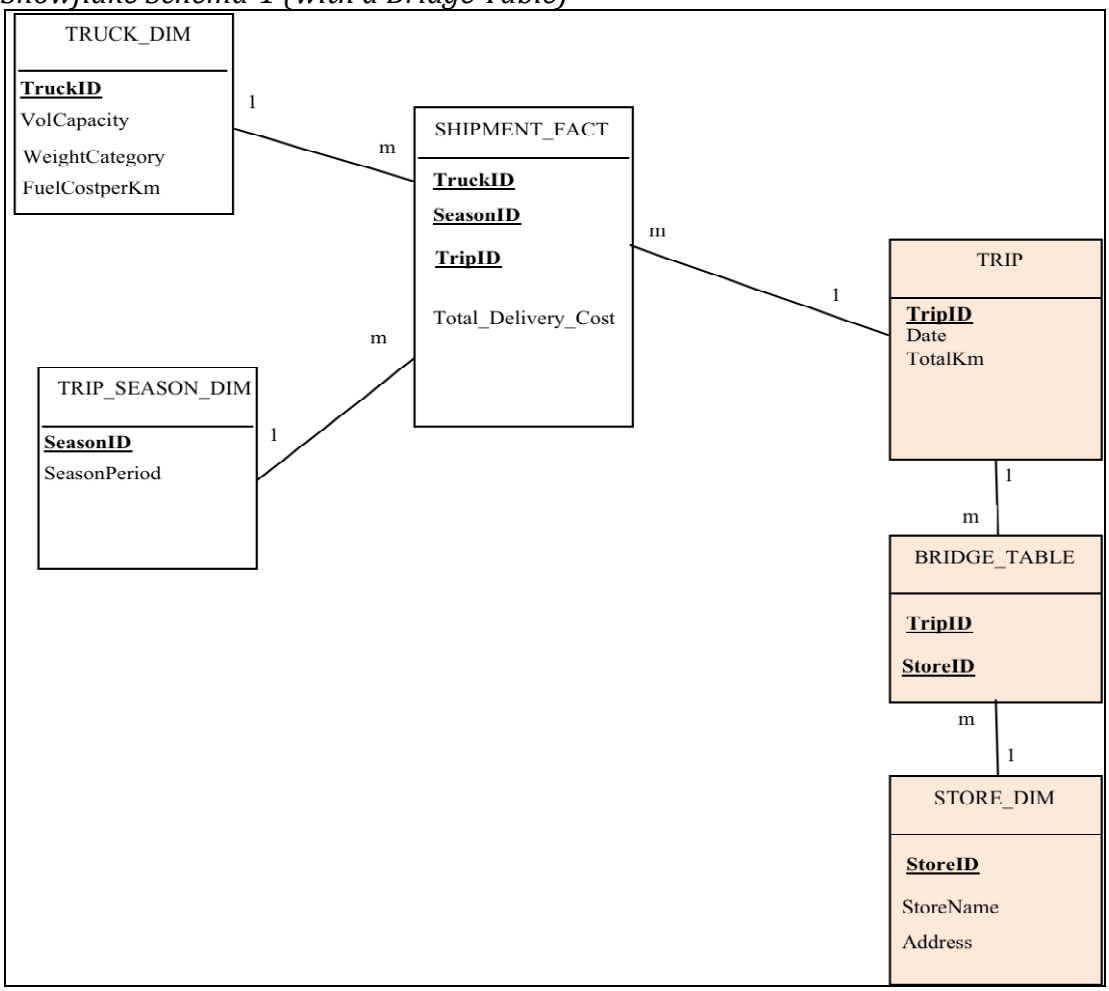
```
Insert Into Store Values ('M3', 'Myer HiPoint', 'High Point');
Insert Into Store Values ('M4', 'Myer West', 'Doncaster');
Insert Into Store Values ('M5', 'Myer North', 'Northland');
Insert Into Store Values ('M6', 'Myer South', 'Southland');
Insert Into Store Values ('M7', 'Myer East', 'Eastland');
Insert Into Store Values ('M8', 'Myer Knox', 'Knox');

Insert Into Destination Values ('Trip1', 'M1');
Insert Into Destination Values ('Trip1', 'M2');
Insert Into Destination Values ('Trip1', 'M4');
Insert Into Destination Values ('Trip1', 'M3');
Insert Into Destination Values ('Trip1', 'M8');
Insert Into Destination Values ('Trip2', 'M4');
Insert Into Destination Values ('Trip2', 'M1');
Insert Into Destination Values ('Trip2', 'M2');
```

2. Solution Model 1 – using a Bridge Table

The measurable fact to be included in the fact table is "Total Delivery Cost", which is calculated by *distance* (total kilometres in the Trip table) and *cost per kilometre* (from the Truck table). The dimensions are Truck, Trip Period and Store.

The snowflake schema for this case study is as follows:

Snowflake Schema-1 (with a Bridge Table)



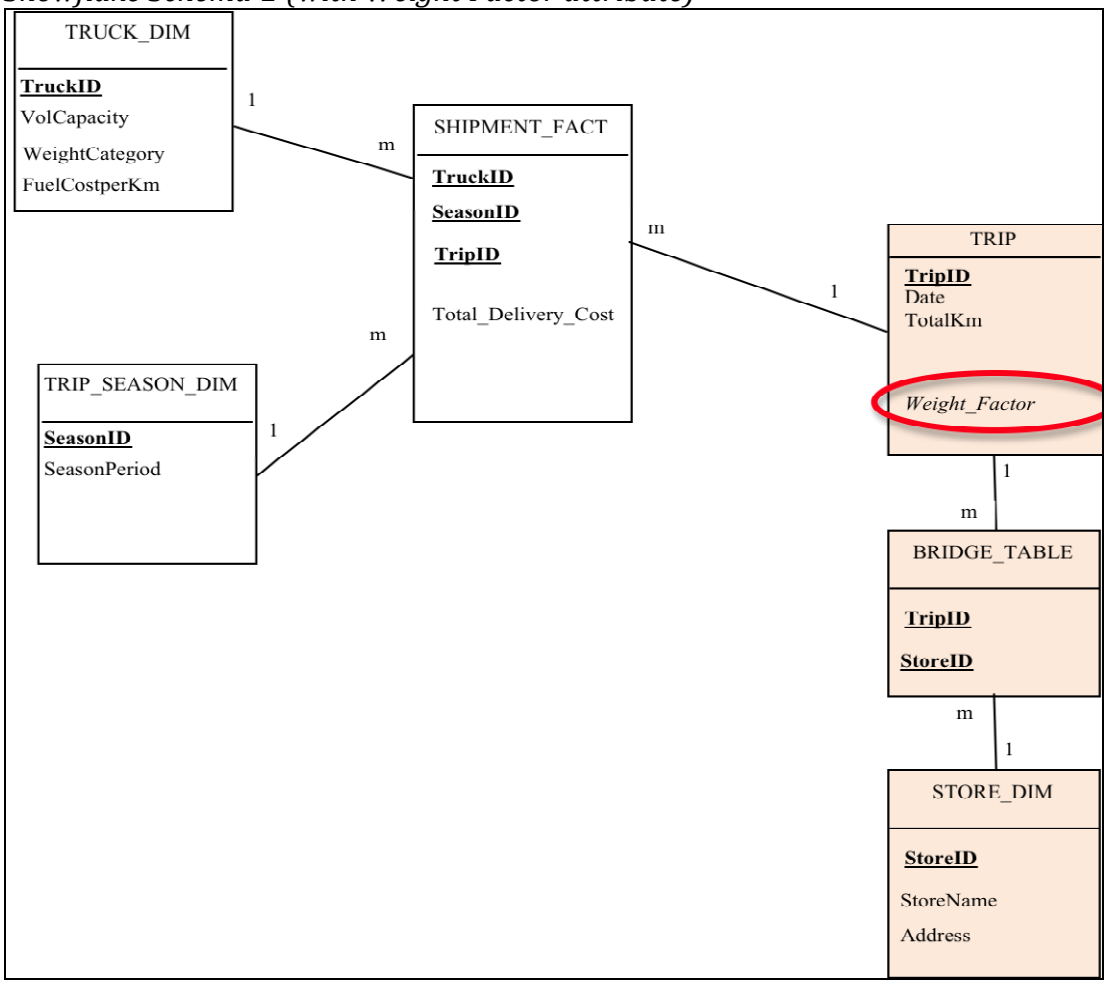
Your tasks:

a. Create a dimension table called TruckDim1.
b. Create a dimension table called TripSeason1. This table will have 4 records (Summer, Autumn, Winter, and Spring).
c. Create a dimension table called TripDim1.
d. Create a bridge table called BridgeTableDim1.
e. Create a dimension table called StoreDim1.
f. Create a tempfact (and perform the necessary alter and update), and then create the final fact table (called it TruckFact1).
g. Display (and observe) the contents of the fact table (TruckFact1).

3. Solution Model 2 – add a Weight attribute in the Bridge

The second data warehouse is shown by the following snowflake schema-2. Notice that a weight factor attribute is added to the Trip dimension.

Snowflake Schema-2 (with Weight Factor attribute)



Your tasks:

a. Create a dimension table called TruckDim2.
b. Create a dimension table called TripSeason2. This table will have 4 records (Summer, Autumn, Winter, and Spring).
c. Create a dimension table called StoreDim2.
d. Create a bridge table called BridgeTableDim2.

Notes: tasks (a)-(d) to create dimension tables version 2, are identical to the previous tasks (a)-(d) to create dimension tables version 1 in the previous section. However, for the sake of completeness of Schema v2, you need to create the dimension tables v2. You can either copy directly from dimension tables v1, or create the dimension tables v2 the same way you did for v1.

e. Create a dimension table called TripDim2 (Notes: this dimension is different from TripDim1 in the previous section).

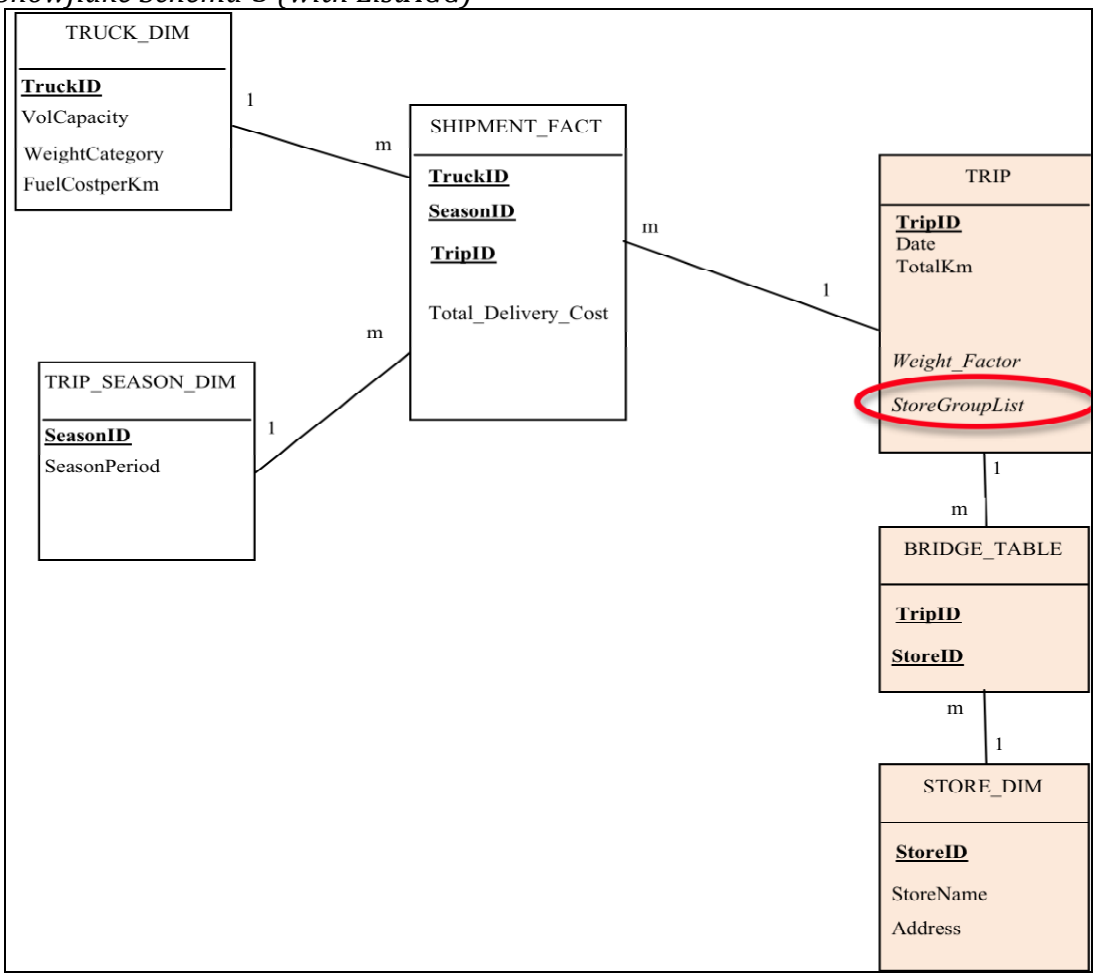
Hints:
(i) Before creating table TripDim2, you need to do a bit of experiments. We don't want to create a table immediately. Instead we are going to try a few Select statements. Once the correct Select statement is identified, then we can use Create Table as Select to create table TripDim2.
(ii) Do a select statement to display the contents of the Trip Table. Then do a select statement do display the contents of the Destination Table. Inspect visually, how many stores that each trip has.
(iii) Now, do another select statement to display the TripID, Date, TotalKm, together with Number of Stores (you need to use an aggregate function to count number of stores per trip). Can you find a trip that does not have any store in the operational database? Is this trip included in the query result? If not, why not?
(iv) Revise the above select statement, so that the number of stores is not displayed as an integer number, but as a percentage. For example, if the number of stores is 5, then it should show 0.2 (20%), instead of 5.
(v) Now, you are ready to create table TripDim2. Don't forget that the column that stores the weight must be named

f. Create a tempfact (and perform the necessary alter and update), and then create the final fact table (called it TruckFact2).
g. Display (and observe) the contents of the fact table (TruckFact2).
h. What is the total delivery cost for each store?

4. Solution Model 3 – A ListAGG version

The third version of the data warehouse is having a ListAgg attribute, which holds the information of the group of each record in the parent dimension table. In this case, the Trip dimension is added with an attribute called StoreGroupList which lists a group of stores for each trip.

Snowflake Schema-3 (with ListAGG)



Your tasks:

a. Create a dimension table called TruckDim3.
b. Create a dimension table called TripSeason3. This table will have 4 records (Summer, Autumn, Winter, and Spring).
c. Create a dimension table called StoreDim3.
d. Create a bridge table called BridgeTableDim3.

Notes: tasks (a)-(d) to create dimension tables version 3, are identical to those for v1 and v2 in the previous section. However, for the sake of completeness of Schema v3, it would be better if you could create the dimension tables for v3 separately.

e. Create a dimension table called TripDim3 (Note that TripDim3 is different from TripDim1 and TripDim2).

Hints:

(i) Like with TripDim2, before creating table TripDim3, we would like to experiment with a few Select statements to find out what is the best way to create table TripDim3.

(ii) Do a select statement to display TripID, TripDate, TotalKm, and WeightFactor.

(iii) Now do a select statement to display TripID, TripDate, TotalKm, WeightFactor, and StoreGroupList. The StoreGroupList column is implemented by the ListAGG function.

The ListAGG function has the following format:
LISTAGG (attr1, '_') Within Group (Order By attr1) As columnname

Attr1 is the store id, and the '_' is to indicate that the store ids are concatenated with the '_' symbol (e.g. M1_M2_M3_M4_M8). If we want to have the stores listed in a descending order (e.g. M8_M4_M3_M2_M1), then we use Within Group (Order By attr1 Desc).

(iv) You can now create table TripDim3 by using the above select statement.

f. Create a tempfact (and perform the necessary alter and update), and then create the final fact table (called it TruckFact3).

g. Display (and observe) the contents of the fact table (TruckFact3).