Laboratory 4
Data Cleaning and Simple Star Schema

Problem Description – Monash Product Online Auction:

Monash Product Online Auction is a new online auction system where users/customers can buy premium products at a fraction of the retail cost. To be able to participate in the auctions, customers need to have bid packages. This bid package can be used to buy an item during the auction. In general, the customer with the highest bid when the time reaches zero wins the auction and buys the product for this price.

The auction system also sells products at a normal retail price, which customers can choose to buy if they lose at the auction. Customers can buy the product at any time when the auction is online and up to 24 hours after the auction is closed. Depending on the maximum refund policy for the item, customers will get the money they have spent on the auction refunded if they decide to buy the product.

Participating in an online auction system is often influenced by the Internet speed and hardware resources on the customer side. Moreover, many customers do not have time to sit down and wait for auctions to complete, and prefer to ask a third party agent to participate in the auction to buy the products they want. To facilitate this need, the online auction system also provides Bid Agents to be used by the customers. Customers can hire a Bid Agent to participate in an auction on their behalf (at no cost), and the customer will inform the agent of their requirements (e.g. maximum price they want to pay for the product, etc).

Currently, the operational database system consists of a list of customers (CUSTOMER), the country and region of the customers (NATION and REGION), a list of items in the auctions (ITEM, BIDPACKAGE), a list of auctions of the company (AUCTIONS), and part of auction activities (BIDDING, BUYAUCTION). These tables can be found at the `dtaniar` database. The operational database tables can be accessed using the following SQL commands:

```
select * from dtaniar.region2;
select * from dtaniar.nation2;
select * from dtaniar.category2;
select * from dtaniar.item2;
select * from dtaniar.bidpackage2;
select * from dtaniar.customer2;
select * from dtaniar.auction2;
select * from dtaniar.buyauction2;
select * from dtaniar.bidding2;
```

Build a Data Warehouse – Your Tasks:

You are required to build a data warehouse for the above Monash Product Online Auction database.

The summary information that must be included in the data warehouse is:

- Total Profit: (auctionPrice/biddingPrice)*singleBidCost - retailPrice
- Total Number of Auctions

Your data warehouse should be able to show information such as:

1. What is the total profit list based on item category and item name?
2. What is the total number of auctions in each season (e.g. Summer, Autum, Winter, Spring) in different regions/nations?

You first need to draw a suitable star schema, and then use the SQL commands to create the data warehouse. You need to make sure that the input data to the data warehouse has undergone a reasonable data cleaning process, in order to make sure that the data in the data warehouse is accurate.

Then, you need to answer the above two queries using the SQL commands. In addition, you need to come up with one more question, answer the question using the SQL commands, and provide with a reason why the management would like to have such information. All the three SQL queries must use the fact table and at least one dimension table.

Additional information and assumptions:

1. The operational database for this case study is a relatively big database, whereas the star schema (data warehouse) you are going to build is small (which is only based on the two questions: "What is the total profit list based on item category and item name?" and " What is the total number of auctions in each season in different regions/nations?"). Therefore, your data warehouse may not need to use all the tables in the operational database. Subsequently, data cleaning is only related to the operational tables that you will be used in the data warehouse. Note that you will only need to clean the data that will affect your data warehouse.

2. Before a customer participates in the bidding process, he/she needs to buy a bidpackage. A bidpackage consists of "coins". Every time, he/she wants to bid, he/she uses "one coin" only. This kind of auctions is different from normal auctions, where you can increase each bid by any dollars. Whereas in this auction, for every bid, you only increase the bidding price by "one coin" only. If "one coin" is equivalent to 2cents, each bid is only increased by 2cents. Every time you bid, it also costs you money (for example 75cents); this means that every time you bid, you need to pay 75cents. So this is what (auctionPrice/biddingPrice) * singleBidCost is. A bidpackage may contain "20coins or 50 coins or 100coins", depending on which bidpackage the customer bought. If the customer bought a "20coin"

bidpackage, then obviously this customer can only bid 20times, before he/she runs out of "bidding coins".

3. The totalProfit is (auctionPrice/biddingPrice) * singleBidCost - retailPrice. This excludes items that are sold through direct buy, rather than through auction. Hence, for simplicity, we can exclude items that are sold through direct buy (we only consider items that are sold through auction). In other words, we will not consider BuyAuction2 table. Also for the biddingPrice and singleBidCost, you may use constant values such as 2cents and 75cents (as hinted in point #2 above), due to incomplete data in the bidpackage2 table.

4. Note that a bidpackage IS ALSO an item. This is why BP_ITEMKEY in bidpackage2 and I_ITEMKEY in item2 share the same domain, since a bidpackage is also an item. Consequently, the company may want to "sell" a bidpackage through an auction, like other normal items.

5. The data used in this case study is obtained by an automatic program crawler. Because of the limitation of the program crawler, the entire bid history for a particular auction is not obtained; however, the winning auction itself is correctly recorded in the system.

6. Due to the limited data obtained by the crawler, not all bid packages are used in this case study. In this case, it may be appropriate to use a constant for the singleBidCost and the biddingPrice.

7. Since the auction is online, its location is considered as the same as the winner (customer)'s location.