# TASK DESCRIPTIONS

## Requirements:

In order to complete the current testing, you will need to load the provided tables’ data (excel files or sql insert statements – on your choice):

* rpm\_future\_retail
* rpm\_zone\_future\_retail
* rpm\_zone\_location

into your oracle database as described below in **Loading Data** section.

**Loading Data:** The provided **tables\_structures.sql** script will help you to understand the data types, create tables and then load in the data by using **only one** of two options below:

1. Preferred option: using sql insert statements from given ***.sql*** files (inside *data\_in\_sql* folder) for each table – just run them once you have created tables.
2. Tricky option: manually, using ***excel*** files (inside *data\_in\_excel* folder). If this option preferred, then feel free to modify the values’ types inside the excel files in case of issues to load them in (e.g. 3.4400 -> 3,4400 – dot to comma separation in case if it cannot be loaded in original state).

## Assumptions:

* The provided file called **RPM\_overview.doc** can help you to better understand the purpose of these tables (optional to read, if not clear the data structure).
* *RPM\_ZONE\_FUTURE\_RETAIL* table will hold zone level retail price changes. (*item/zone/action\_date* unique values)
* *RPM\_FUTURE\_RETAIL* table will hold retail price information (historical and future) for *items/locations*, and this table can hold only one *item/location/action\_date* combination, otherwise they are duplicates.
* *RPM\_ZONE\_LOCATION* table maps *zones* and *locations*, so, if given *zone\_id* one can always retrieve *locations* that are linked to it. Any location can be linked to only one pricing zone.
* In our configuration, all the locations within one zone should have the same prices for item.

## What we expect:

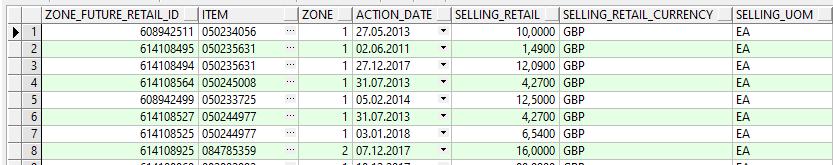
The result of each task is a SQL query (DML/DDL) that will be used to estimate your knowledge.

If you can solve partially, and not the whole exercise, then do so, and we will estimate accordingly. Please write all your concerns and assumptions, if you have any, to support your decisions.

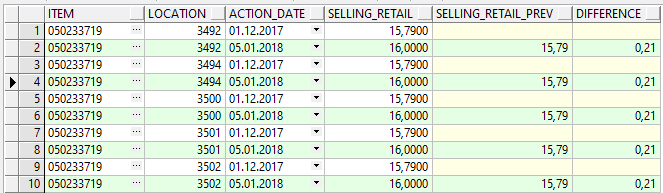
**Good luck!**

## TEST QUESTIONS:

1. Find duplicate records in RPM\_FUTURE\_RETAIL. The record is a duplicate if there already such *item/location/action\_date* combination exists (prices can differ):
   1. Write a query to return all duplicated records.
   2. Delete duplicate records, so that *item/location/action\_date* will be unique.
2. Write a query to find all item/zone combinations in *rpm\_zone\_future\_retail* for which there are no pricing data exists at the location level (*rpm\_future\_retail*). **Note** that locations linked to each zone can be obtained from *rpm\_zone\_location* table (e.g.: zone: 1 = locations: 2302, 3040, … etc). Below is an example of the result set:



1. Write a query that will return current and previous selling retail prices (preceding *action\_date*) for each *item/location* combination and the difference between current and previous prices. Below is an example of the result set:



**Hint:** you might consider (not mandatory) using LEAD/LAG function to retrieve previous prices and CASE function to compute the differences if previous prices exist, otherwise, return null or 0 on your choice (**optional**).

1. Assuming that the current date in the system is today
2. Find the price for each item at each location on the current date. Since there are many prices for the same *item/location* combination, use the *selling\_retail* values closest to the current date, meaning latest *action\_date* which is <= current date).
3. find the differences in the prices between zone 1 and 2 for all items on the current date. See below result set example:

