Database User Guide

## Database setup

1. Download and unzip Team2\_eggshell.zip.
2. SQLDeveloper Database Connection:
   1. Authentication:

|  |  |
| --- | --- |
| Username: | Team2 |
| Password: | Password99 |

* 1. Details:

|  |  |
| --- | --- |
| Hostname: | Heinz-db-03.andrew.local.cmu.edu |
| Port: | 1561 |
| Service\_name: | https://heinzdb3.andrew.cmu.edu/ |

1. Open the Codes folder and edit main.sql by changing the base\_path to the directory of the Codes folder ending with /.

e.g. '/Users/chantimyeung/Dropbox/Mac/Desktop/Codes/';

1. At the SQL> prompt, type in start (or @)<pathname>\main.sql hit enter, which will create database tables and objects in order.

## Applications in solving business problems

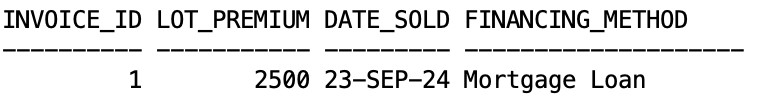
### Scenario 1: Restricted database access for customers

In SQL Developer execute <pathname>\scene1.sql

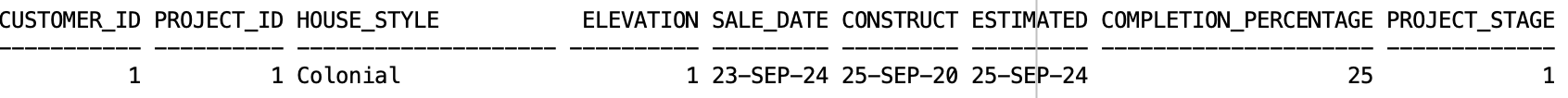
Based on the case study requirement and our assumption, Customers are accessible to Stage 1, 4 and 7 only and the classification of stages follows:

* Stages 1 to 3 are shown as Stage 1
* Stages 4 to 6 are shown as Stage 4
* Stage 7 is shown as Stage 7

In the test case, we are connected as a customer with customer\_lname of TEAM2 and granted CustomerRole. Then, we can perform a sales record check, which is only restricted to our USER id, which is assumed to be the customer's last name. The expected output of CustomerSaleRecord:

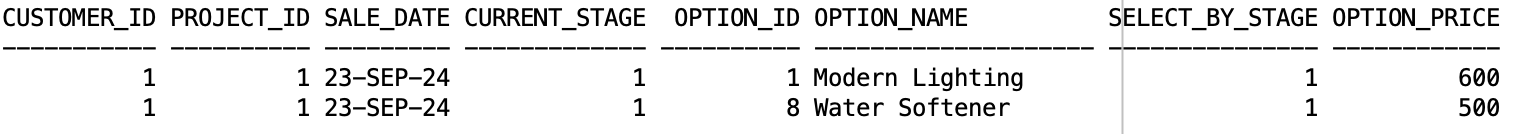


We can also check our project progress via ACCESSIBLE\_PROJECTS view and the expected output as follows:



(\*Note that the Project\_Stage is displayed as 1 while the actual project stage inserted in the Construction\_Project table is 2)

Based on the current project stage, we can filter and view the options available by PROJECT\_OPTIONS\_VIEW. The expected output:



### Scenario 2: Sale cancellation

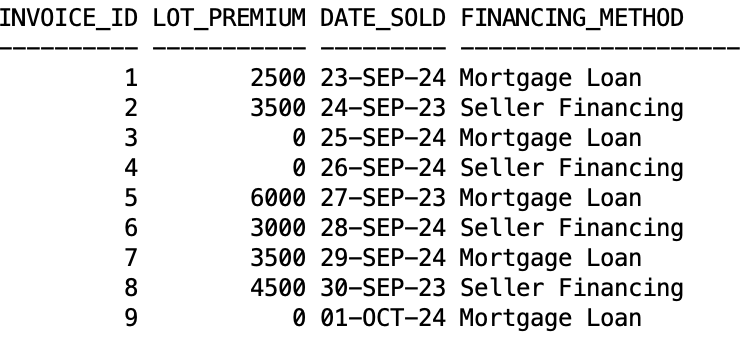
In SQL Developer execute <pathname>\scene2.sql

Based on the Entity Relationship Diagram (ERD) design, the Sale entity forms composite relationships with the entities: Chosen\_Style, Escrow, Construction\_Project. Before a sale record is deleted, a delete cascade is triggered to ensure that related records in Construction\_Project, Escrow, and Chosen\_Style linked to the deleted sale are removed.

According to the case study description and our assumptions, customers’ contract term is effective within one year after it is signed and they can reclaim part of the escrow deposit based on the rule listed in the business rules document. This is performed by the calc\_refund function from the sale\_operations package.

scene2.sql performs the deletion of a sale record with invoice\_id = 1 by calling cancel\_sale\_record procedure and contrasts the Sale tables before and after the deletion. The expected output is as follows.

Extract of the sale table before deletion:

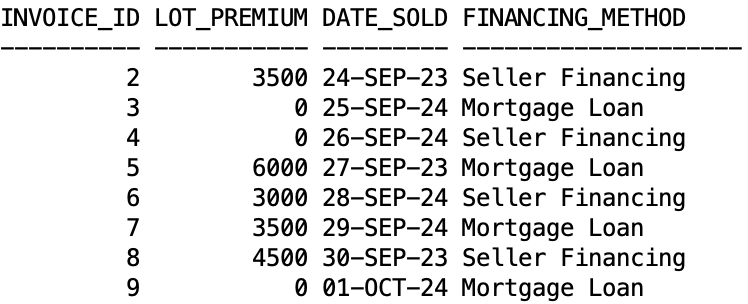


Escrow refund:



(Note that the escrow deposit associated with the deleted sale record is 2000 and the percentage of project completed is 25%. The amount of refund is calculated as: 2000 \* 0.25)

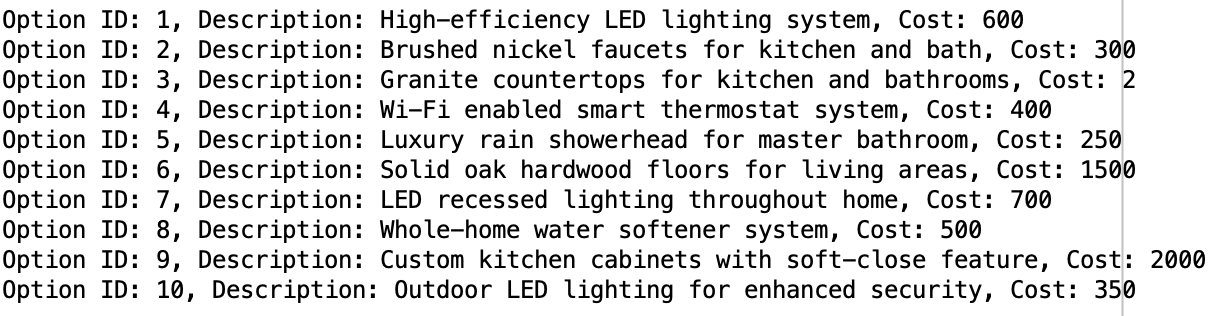
Extract of the sale table after deletion:



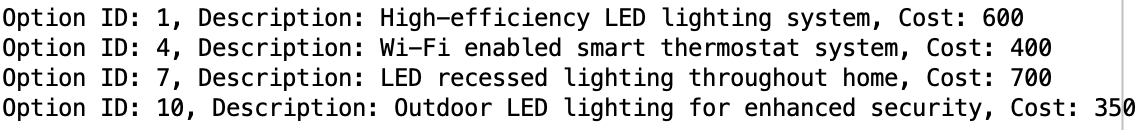
### Scenario 3: Filtering options by category

In SQL Developer execute <pathname>\scene3.sql

Users can sort and filter the Option\_List table by category by using FilterOptionsByCategory procedure, which takes category (e.g. 'Electrical', 'Exterior', 'Interior', 'Plumbing') as the input and outputs the respective information. By inputting NULL, all the option choices on the list will be displayed as follows:



If 'Electrical' is input, the expected output follows:

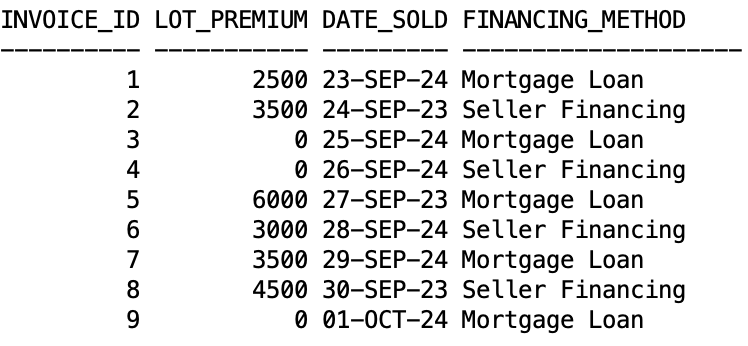


### Scenario 4: Sale record creation

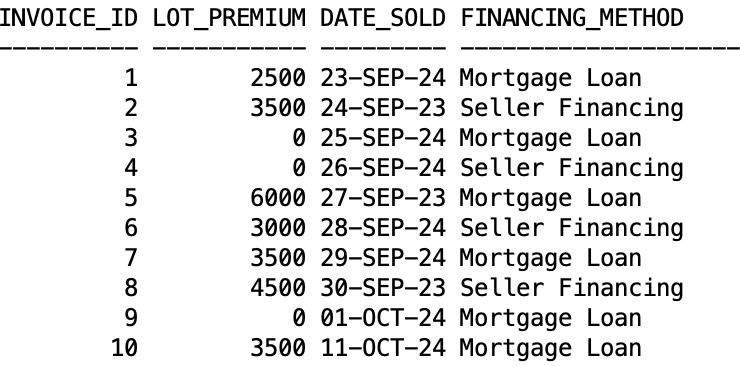
In SQL Developer execute <pathname>\scene4.sql

Users can create a sale record by using create\_sale\_record procedures with necessary parameters from the sale\_operations package.

Extract of the sale table before record creation:



Extract of the sale table after record creation:

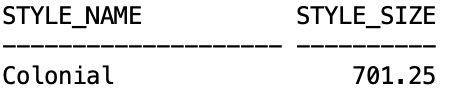


### Scenario 5: Style size derivation

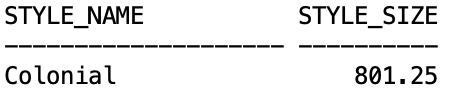
In SQL Developer execute <pathname>\scene5.sql

The style\_size attribute of a House\_Style entity is derived from the summation of the associated room\_size in the Room table. This test inserts a new room entry to the Room table to trigger update\_style\_size.

The size of the Colonial style:



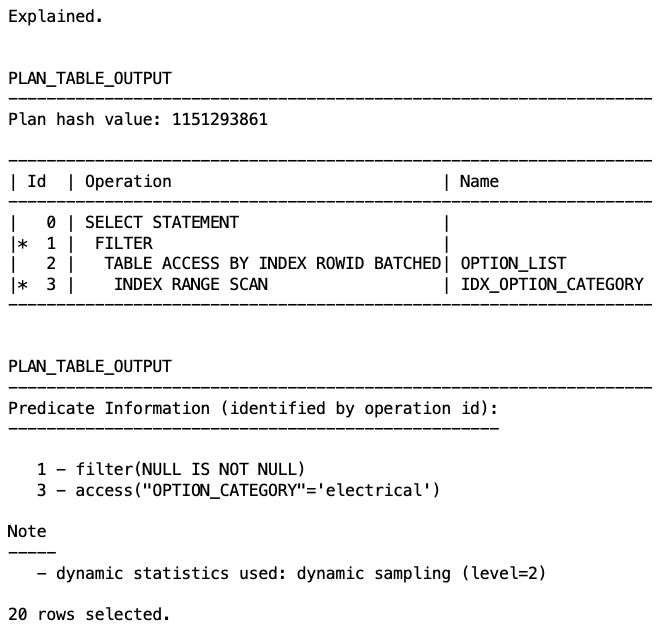
After a new room with size of 100 is added for the Colonial style, the new size follows:

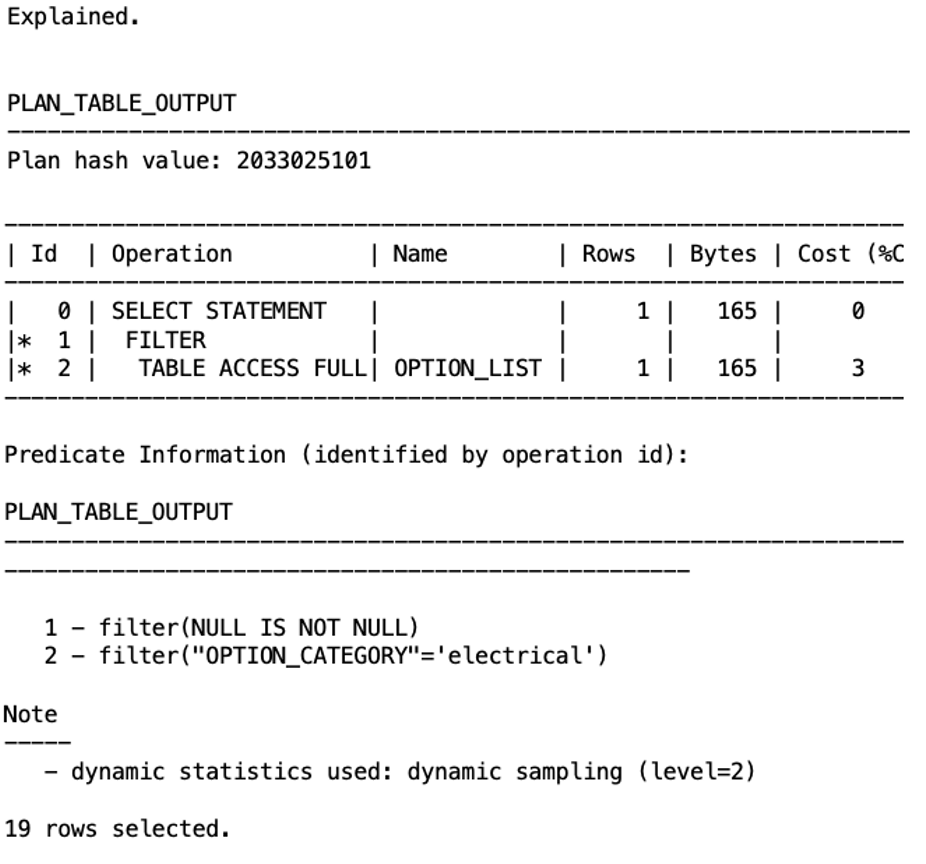


### Scenario 6: Alternative Index

In SQL Developer execute <pathname>\scene6.sql

This test evaluates differences in performance after creating an alternate index on OPTION\_CATEGORY for the OPTION\_LIST table. The first test evaluates the query performance on the Option\_List table by comparing the effects of indexing. First, it runs an EXPLAIN PLAN for a query that filters by the `option\_category` column. Then, the execution plan is displayed using DBMS\_XPLAN.DISPLAY. In the second test, a /\*+ FULL(option\_list) \*/ hint forces a full table scan, bypassing the index to show how the query performs without optimization. The results allow for comparison of query efficiency with and without the index. Expected output:





### Scenario 7: Test Scheduled Job

In SQL Developer execute <pathname>\scene7.sql

This scene inserts 4 new customers that have no sale to remove. Inside this test, it disables the scheduled job, sets the start time to now and re-enables the job. This scheduled job clears all customers who have not purchased a lot yearly.

