Buckfast Abbey

Database System Proposal

[Ruslan Kalashnikov](mailto:kalashnikovr@my.lanecc.edu) & Joseph Sepe

CS276

Winter, 2021

**Abstract**

This project details a database system proposal for Buckfast Abbey to implement and streamline it’s day-to-day operations. The resulting relational database tackles business-related aspects of the Abbey such as guest house room and restaurant reservations, lay employee and monk shift scheduling, as well as gift shop, bookstore and candy shop transactions.

The contents of this proposal detail the progression of the iterative process of the database creation starting with entities list and culminating with the physical design and use case implementation.

# TABLE OF CONTENTS

[**TABLE OF CONTENTS**](#_gfe1son43z11) **2**

[**Entities List**](#_g034zr6zf7gn) **3**

[**Business Rules**](#_8myglrx6duzb) **4**

[**Conceptual ERD**](#_nubngeqmd8h7) **6**

[**Logical ERD**](#_o67fe8ubmqug) **7**

[**Data Warehouse**](#_1imorf2m85mp) **8**

[Potential Use Cases](#_g6el1l3uzte1) 8

[**Physical Implementation**](#_7kko56y5dqmb) **9**

[**Use Case Code**](#_eh965fc3ymx9) **10**

[Use Case 1](#_60hjop7qk2w0) 10

[Use Case 2](#_vow3iqlb0alr) 11

[Use Case 3](#_r5jnrl50og5a) 12

[Use Case 4](#_7tw7ph5j8o51) 13

[Use Case 5](#_jlybqcjxwai) 14

[Use Case 6](#_l82mdzo8cbar) 15

[Use Case 7](#_l2nurtlz2rox) 18

[Use Case 8](#_x4twtpqe9nox) 19

[Use Case 9](#_50miq78w6naj) 20

[Use Case 10](#_4jrfw3vxdrop) 25

[**Index Choice**](#_cpaj1it5sel9) **28**

# Entities List

**Visitor**

**Transaction**

**Product**

**Shop**

**Restaurant**

**House**

**Room**

**Reservation**

**Monk**

**Employee**

**Time Card**

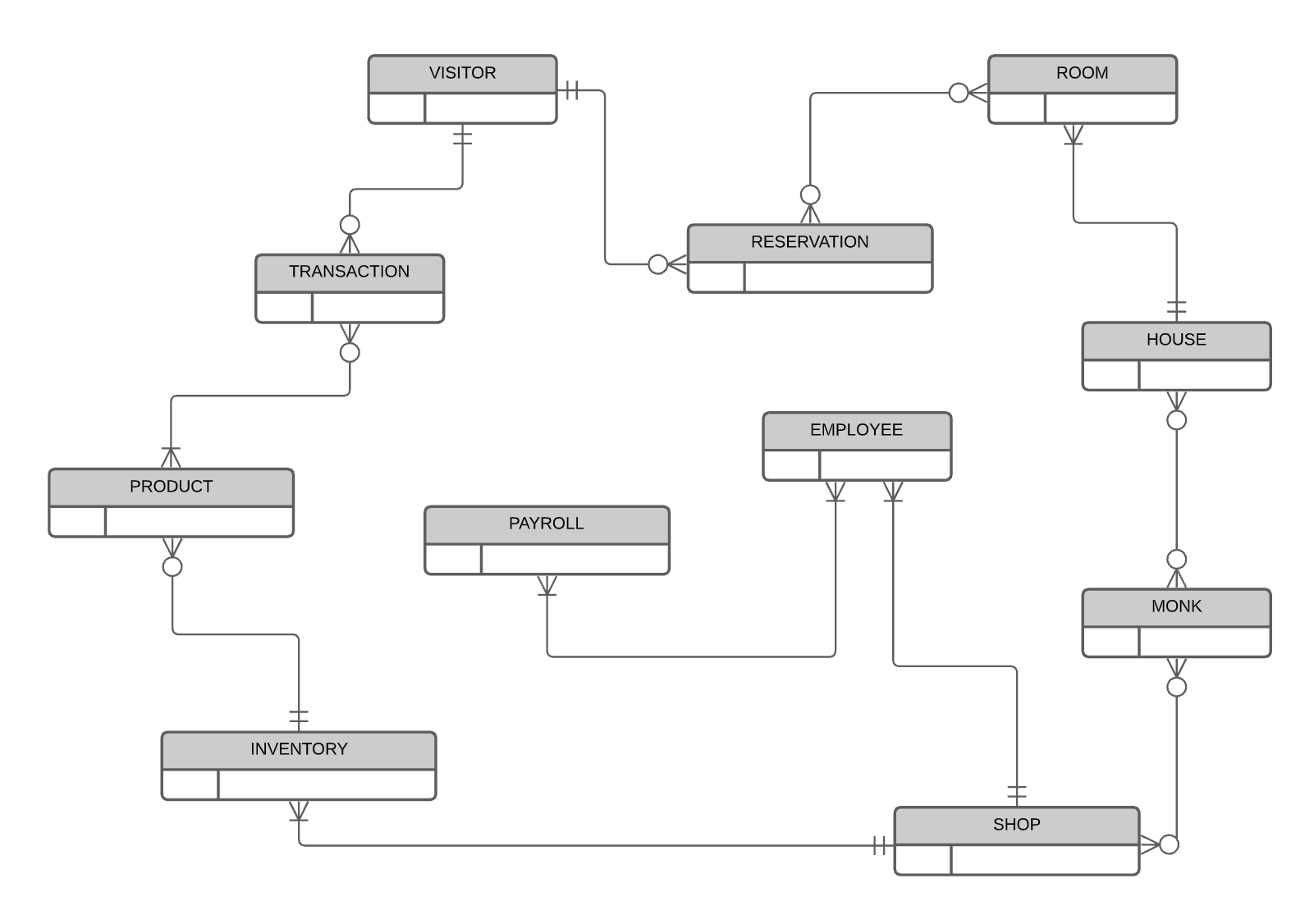
**Payroll**

**Inventory**

# Business Rules

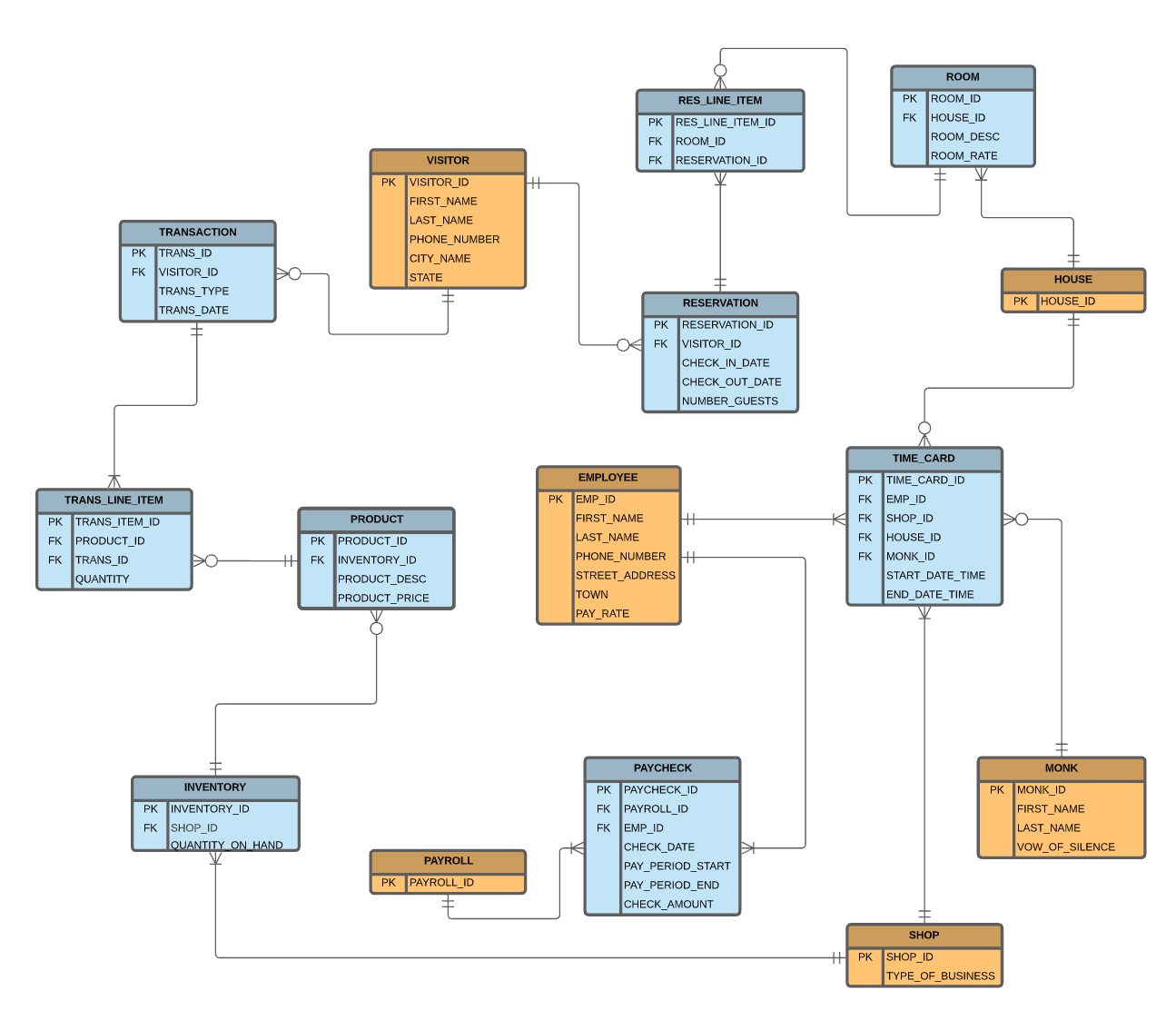
1. A **VISITOR** may make 0 to many **RESERVATIONS**; A **RESERVATION** must be made by 1 and only 1 **VISITOR**.
2. A **RESERVATION** may be for 0 to many **ROOM**s; A **ROOM** may belong to 0 to many **RESERVATION**s.
3. A **HOUSE** must contain 1 to many **ROOM**s; A **ROOM** must belong to 1 and only 1 **HOUSE**.
4. A **VISITOR** may have 0 to many **TRANSACTION**s; A **TRANSACTION** must be made from 1 and only 1 **VISITOR**.
5. A **VISITOR** may purchase 0 to many **PRODUCT**s; A **PRODUCT** may be purchased by 0 to many **VISITOR**s.
   1. A **VISITOR** may have 0 to many **TRANSACTION**s; A **TRANSACTION** must belong to 1 and only 1 **VISITOR**.
   2. A **TRANSACTION** will contain 1 to many **PRODUCT**s; A **PRODUCT** will be a part of 0 to many **TRANSACTION**s.
      1. A **PRODUCT** may be on 0 to many **TRANSACTION LINE ITEM**s; A **TRANSACTION LINE ITEM** has 1 and only 1 **PRODUCT**.
      2. A **TRANSACTION** must have 1 to many **TRANSACTION LINE ITEM**s; A **TRANSACTION LINE ITEM** must have 1 and only 1 **TRANSACTION**.
6. A **PRODUCT** will belong to 0 to many **SHOP**s; A **SHOP** may have 0 to many **PRODUCT**s.
   1. A **SHOP** must have 1 to many **INVENTORY**; An **INVENTORY** must belong to 1 and only 1 **SHOP**.
   2. A **PRODUCT** will be a part of 1 and only 1 **INVENTORY**; An **INVENTORY** may contain 0 to many **PRODUCT**s;
7. An **EMPLOYEE** must work at 1 and only 1 **SHOP**; A **SHOP** must employ 1 to many **EMPLOYEE**s.
   1. A **TIME CARD** must belong to 1 and only 1 **EMPLOYEE**; An **EMPLOYEE** must have 1 to many **TIME CARD**s.
   2. A **SHOP** must have 1 to many **TIME CARD**s; A **TIME CARD** must have 1 and only 1 **SHOP**.
8. An **EMPLOYEE** must have 1 to many **PAYROLL**; A **PAYROLL** must have 1 to many **EMPLOYEE**s.
   1. An **EMPLOYEE** must get 1 to many **PAYCHECK**s; A **PAYCHECK** must be sent to 1 and only 1 **EMPLOYEE**;
   2. A **PAYROLL** must process 1 to many **PAYCHECK**s; A **PAYCHECK** must be processed by 1 and only 1 **PAYROLL**.
9. A **MONK** may work at 0 to many **HOUSE**s; A **HOUSE** may have 0 to many **MONK**s.
   1. A **TIME CARD** must have 1 and only 1 **MONK**; A **MONK** may have 0 to many **TIME CARD**s.
   2. A **TIME CARD** must be for 1 and only 1 **HOUSE**; A **HOUSE** may have 0 to many **TIME CARD**s.
10. A **MONK** may work at 0 to many **SHOP**s; A **SHOP** may have 0 to many **MONK**s.
    1. A **TIME CARD** must have 1 and only 1 **MONK**; A **MONK** may have 0 to many **TIME CARD**s.
    2. A **TIME CARD** must be for 1 and only 1 **SHOP**; A **SHOP** may have 0 to many **TIME CARD**s.

# Conceptual ERD



# 

# Logical ERD



# Data Warehouse

## Potential Use Cases

1. What is the average number of items purchased by each customer?
2. How many people pay with a credit card and how many people pay cash?
3. During what hour of the day does the store make the most sales?
4. What is the most popular product sold in the store?
5. What is the least popular product sold in the store?
6. What is the total amount of all purchases made today?
7. What day does the store make the most amount of money?
8. How much money did the store make this week compared to this week last year?

# Physical Implementation

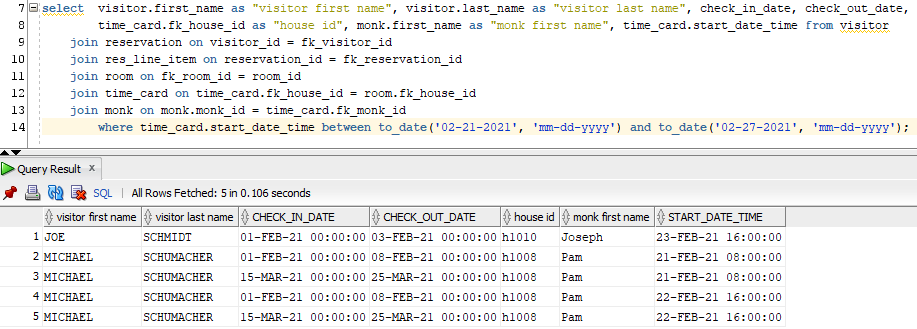
Refer to our [PHYS\_DESIGN.sql](https://github.com/sepej/CS276_Term_Project/blob/main/PHYS_DESIGN.sql) file.

# 

# Use Case Code

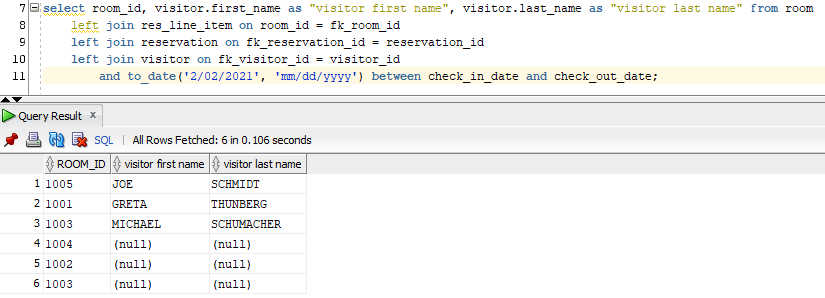
Refer to our [USE\_CASE\_XX.sql](https://github.com/sepej/CS276_Term_Project/tree/main/Use%20Case%20Code) file.

## Use Case 1

A Visitor makes a reservation for February 21, 2021 to February 27, 2021 for one of the Guest Houses, and a monk must be assigned for each of those weeks. Using one query, display the Visitor name, the Guest house, and the monk assigned for each week.

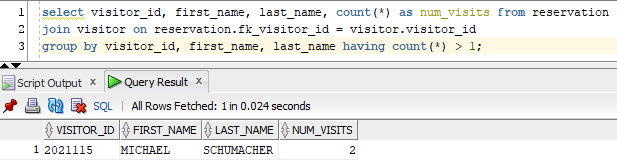
## Use Case 2

The Abbott wants to know all room numbers, as well as those customer names that are staying at the monastery on Friday, February 2, 2021. Not all rooms will be filled…



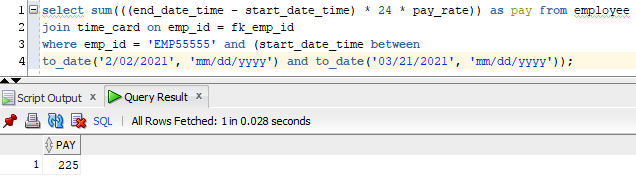
## Use Case 3

The Abbott wants to know if there are any visitors that have made repeat visits (MORE THAN ONE) so they can direct marketing efforts toward them. This query will require an aggregate.

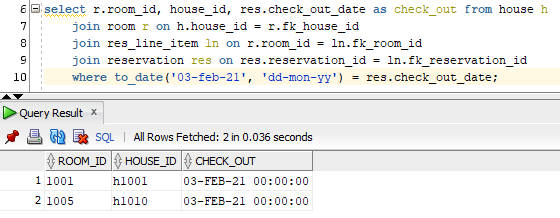


## Use Case 4

The Abbott wants to know the total pay for employee EMP55555 between Feb 2, 2021 and March 21, 2021. You must calculate the number of hours (based on start and end times of each of their shifts), and using the hourly pay, calculate the total pay.



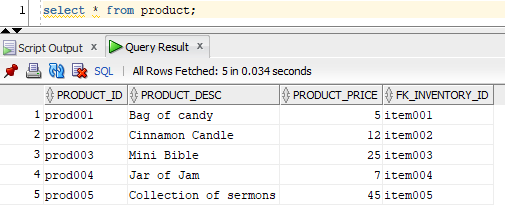
## Use Case 5

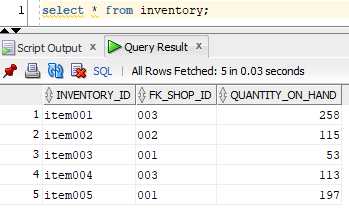
A Monk needs to find out which guest rooms are empty (the guests have checked out) on February 3, 2021 so he can clean them.

## Use Case 6

A vendor delivers a product to the abbey to one of the stores. Write a stored procedure to take in the product ID and a delivery amount and price and insert it into the database if new, or update it if it is currently sold. This will require updating the inventory count. Make sure you test both conditions.

Before:

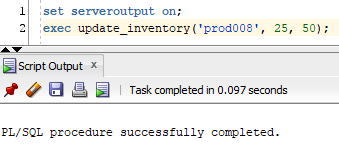


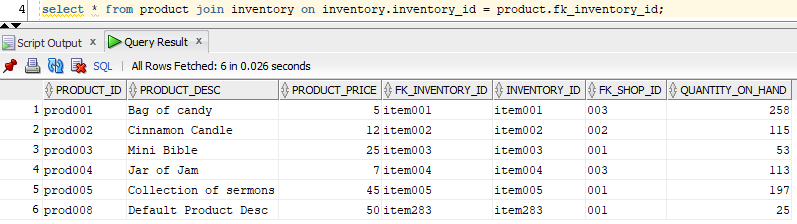


Our procedure:

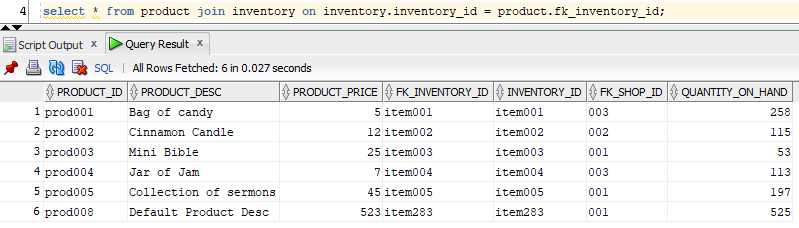
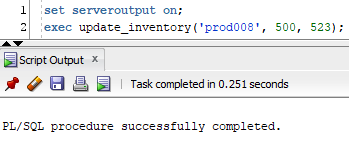


After inserting a new product:



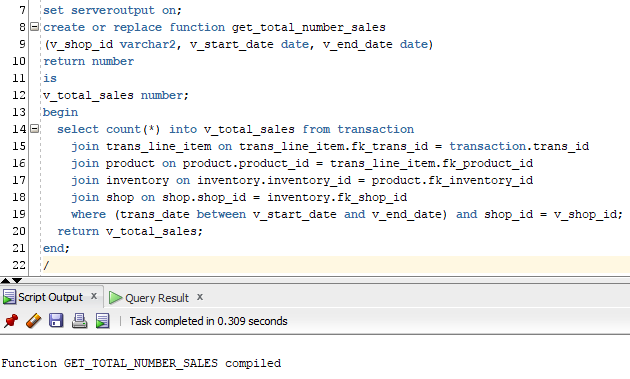


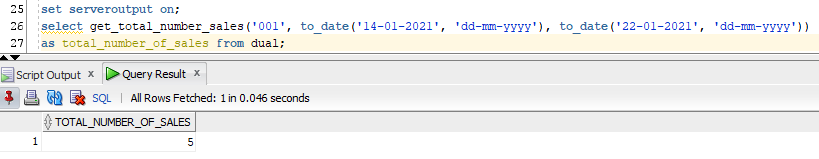
After updating a product:



## Use Case 7

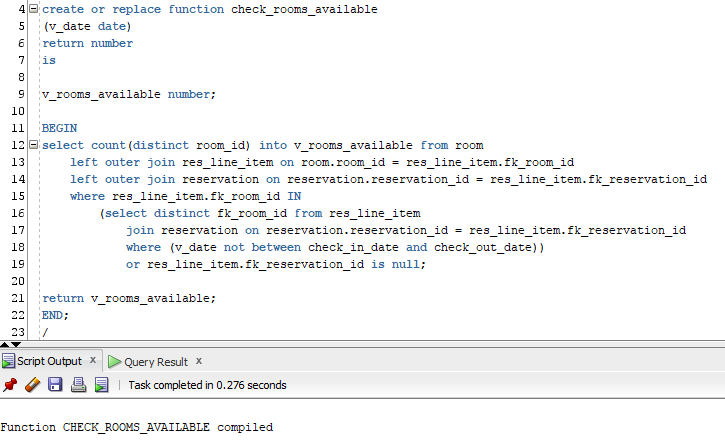
Write a function that will display the total number of sales for one of the stores between two dates. I should be able to call the function passing in a begin date and an end date and store code.

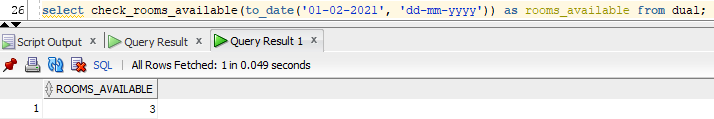




## Use Case 8

Write a function that will return the count of rooms that do NOT have reservations at the Abbey on Saturday, June 8.

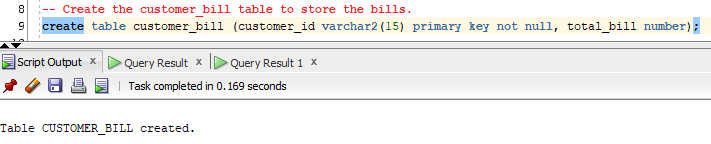




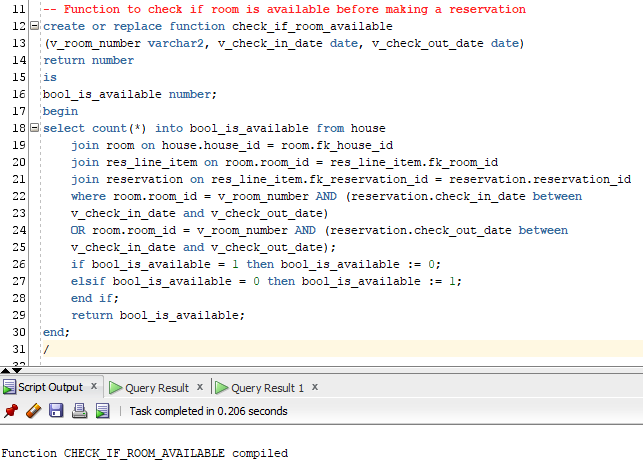
## Use Case 9

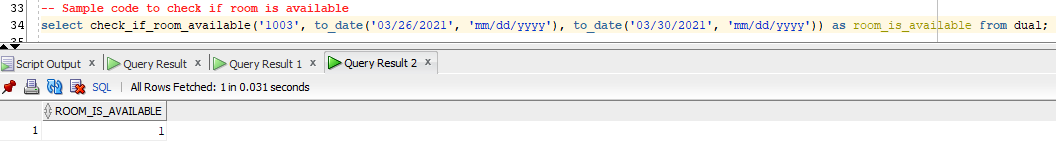
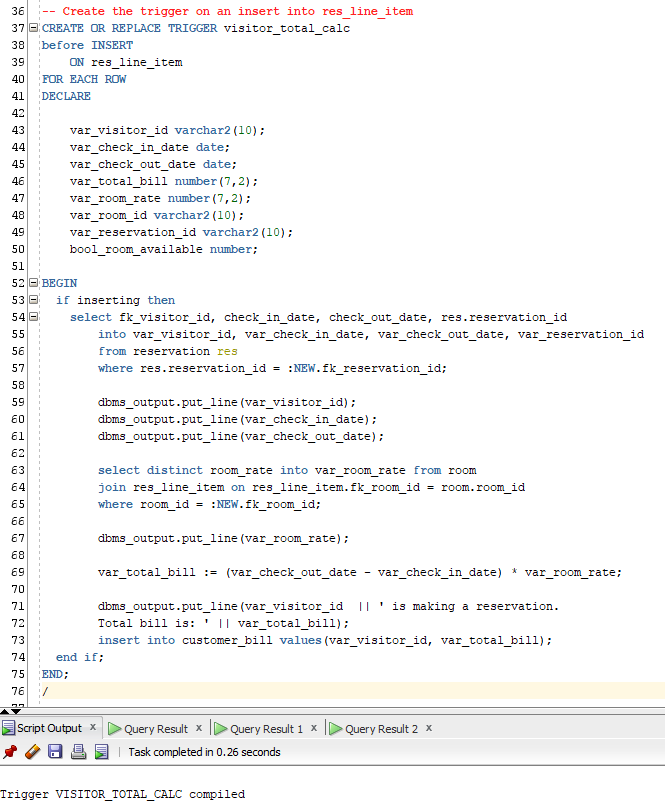
Write a trigger that will calculate a customer’s total Guest House bill and store it in a separate table when they make a reservation at a Guest House. The customer will stay multiple days.

Create the customer\_bill table:

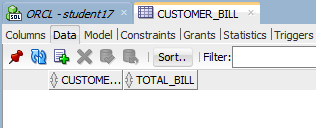


Function to check if the room is available:

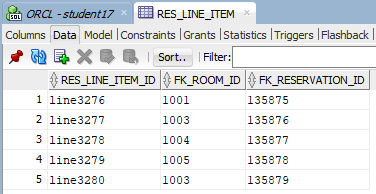


Select statement to see if the room is available:Trigger on an insert into the res\_line\_item table:

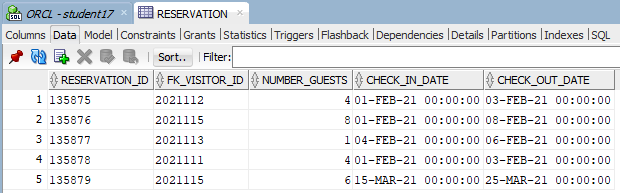
Empty customer\_bill table:



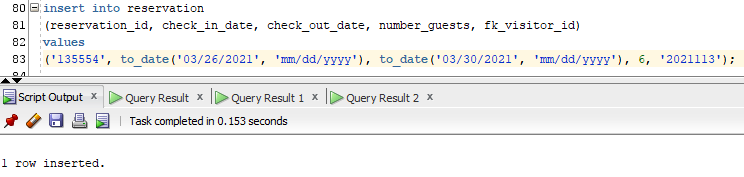
res\_line\_item table before:



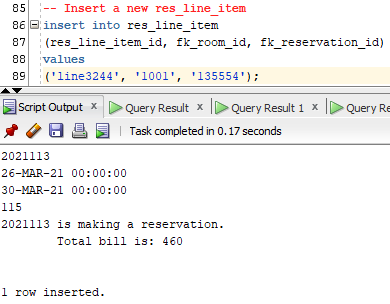
reservation table before:



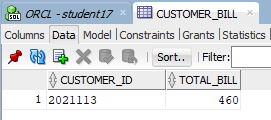
Insert a new reservation:

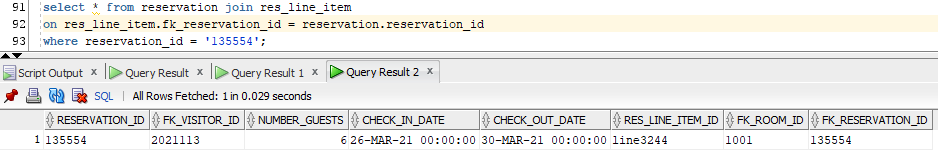


After inserting a new res\_line\_item:



customer\_bill table after the insert:



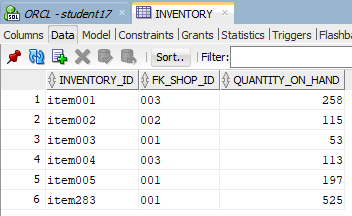
reservation and res\_line\_item table after the inserts:

## Use Case 10

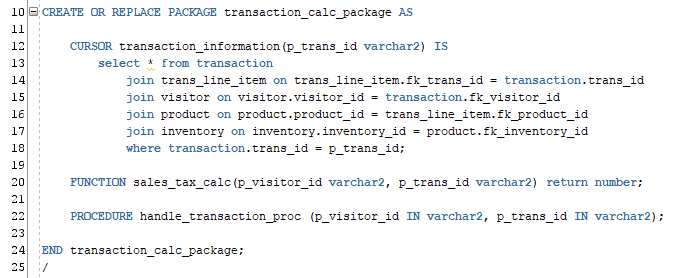
Write a package made up of the following:

1. A function that will calculate the amount of sales tax based on a purchase from Buckfast Abbey, depending on the state where the purchaser is located
2. A procedure which will subtract from inventory
3. Print out the customer id, the first and last names of the customer, the purchase date, and the total cost of the purchase (including the calculated sales tax).

Inventory table before writing package:



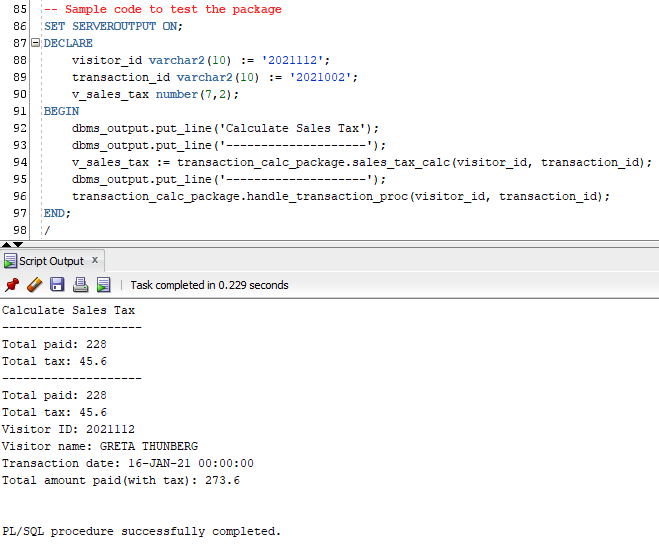
Package spec:



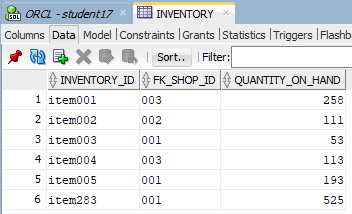
Package body:



Running the package:



Inventory after running the package (item001 and item005 updated):



# Index Choice

1. The index is created on the TIME\_CARD table, employee’s start date time to facilitate retrieval of all of the time card records of employees working that day.

**create index ind\_loc\_to\_employee\_by\_start\_date\_time**

**on time\_card(start\_date\_time);**

1. The index is created on the RESERVATION table, check-in date in order to quickly find all the reservations based on a specific check-in date.

**create index ind\_loc\_to\_reservation\_by\_check\_in\_date on reservation(check\_in\_date);**