Buckfast Abbey

Database System Proposal

Ruslan Kalashnikov & 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

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.

    a. A **VISITOR** may have 0 to many **TRANSACTION**s; A **TRANSACTION** must belong to 1 and only 1 **VISITOR**.

    b. A **TRANSACTION** will contain 1 to many **PRODUCT**s; A **PRODUCT** will be a part of 0 to many **TRANSACTION**s.

        i. A **PRODUCT** may be on 0 to many **TRANSACTION LINE ITEM**s; A **TRANSACTION LINE ITEM** has 1 and only 1 **PRODUCT**.

        ii. 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.

    a. A **SHOP** must have 1 to many **INVENTORY**; An **INVENTORY** must belong to 1 and only 1 **SHOP**.

    b. 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.

    a. A **TIME CARD** must belong to 1 and only 1 **EMPLOYEE**; An **EMPLOYEE** must have 1 to many **TIME CARD**s.

    b. 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.

    a. An **EMPLOYEE** must get 1 to many **PAYCHECK**s; A **PAYCHECK** must be sent to 1 and only 1 **EMPLOYEE**;

    b. 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.

    a. A **TIME CARD** must have 1 and only 1 **MONK**; A **MONK** may have 0 to many **TIME CARD**s.

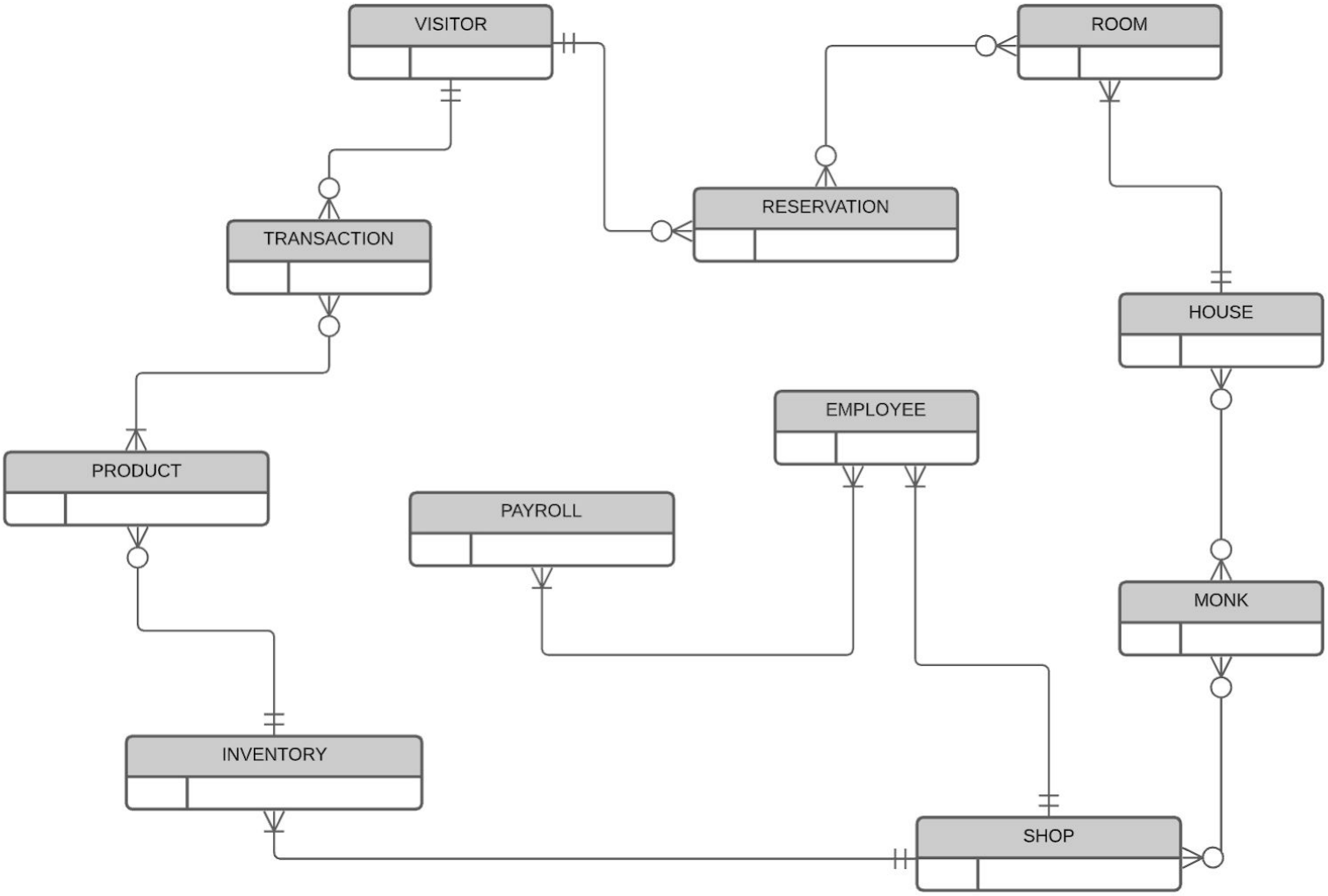
    b. A **TIME CARD** must be for 1 and only 1 **HOUSE**; A **HOUSE** may have 0 to many **TIME CARD**s.

10. A **MONK** may work at 0 to many **SHOP**s; A **SHOP** may have 0 to many **MONK**s.

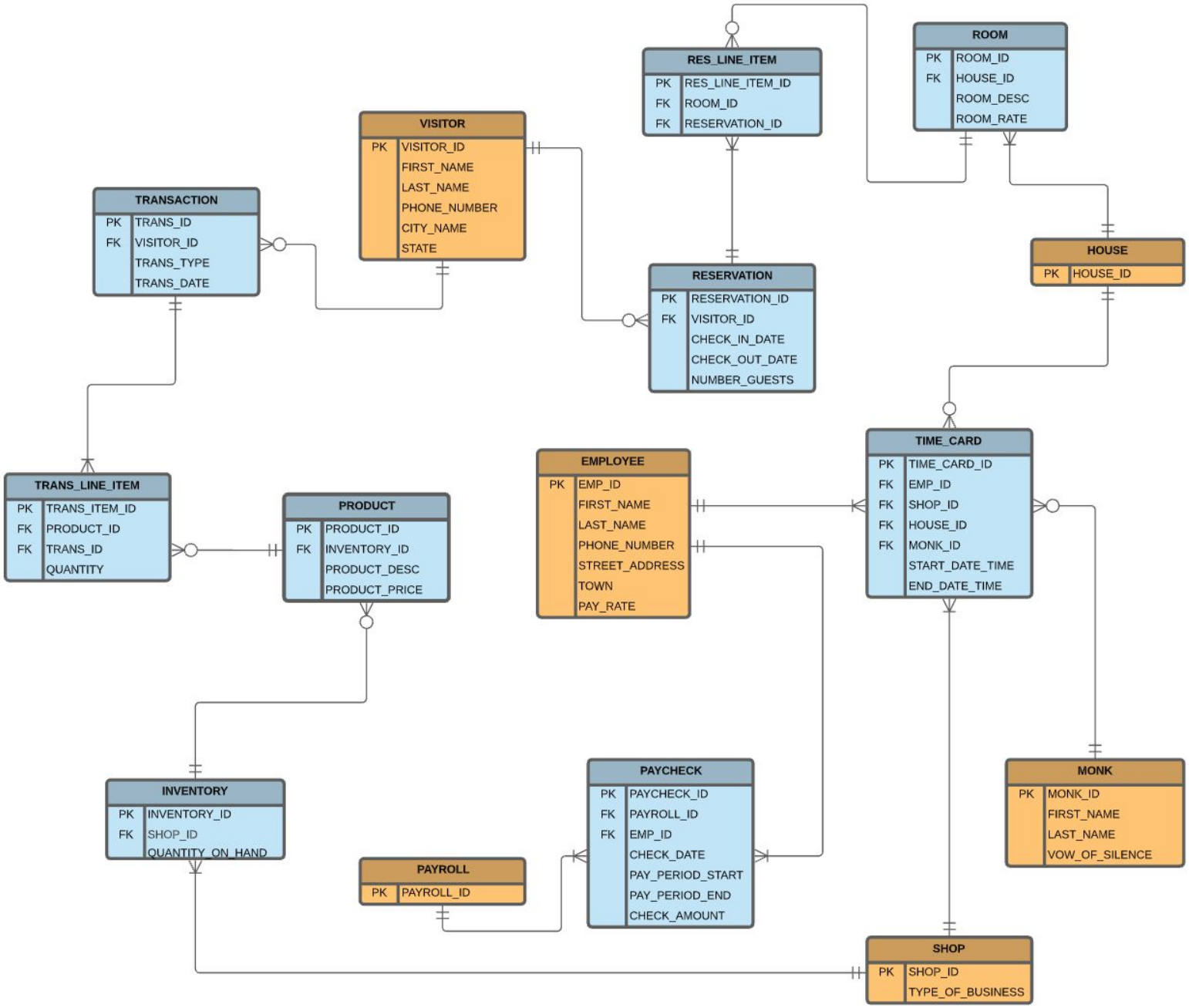
    a. A **TIME CARD** must have 1 and only 1 **MONK**; A **MONK** may have 0 to many **TIME CARD**s.

    b. A **TIME CARD** must be for 1 and only 1 **SHOP**; A **SHOP** may have 0 to many **TIME CARD**s.

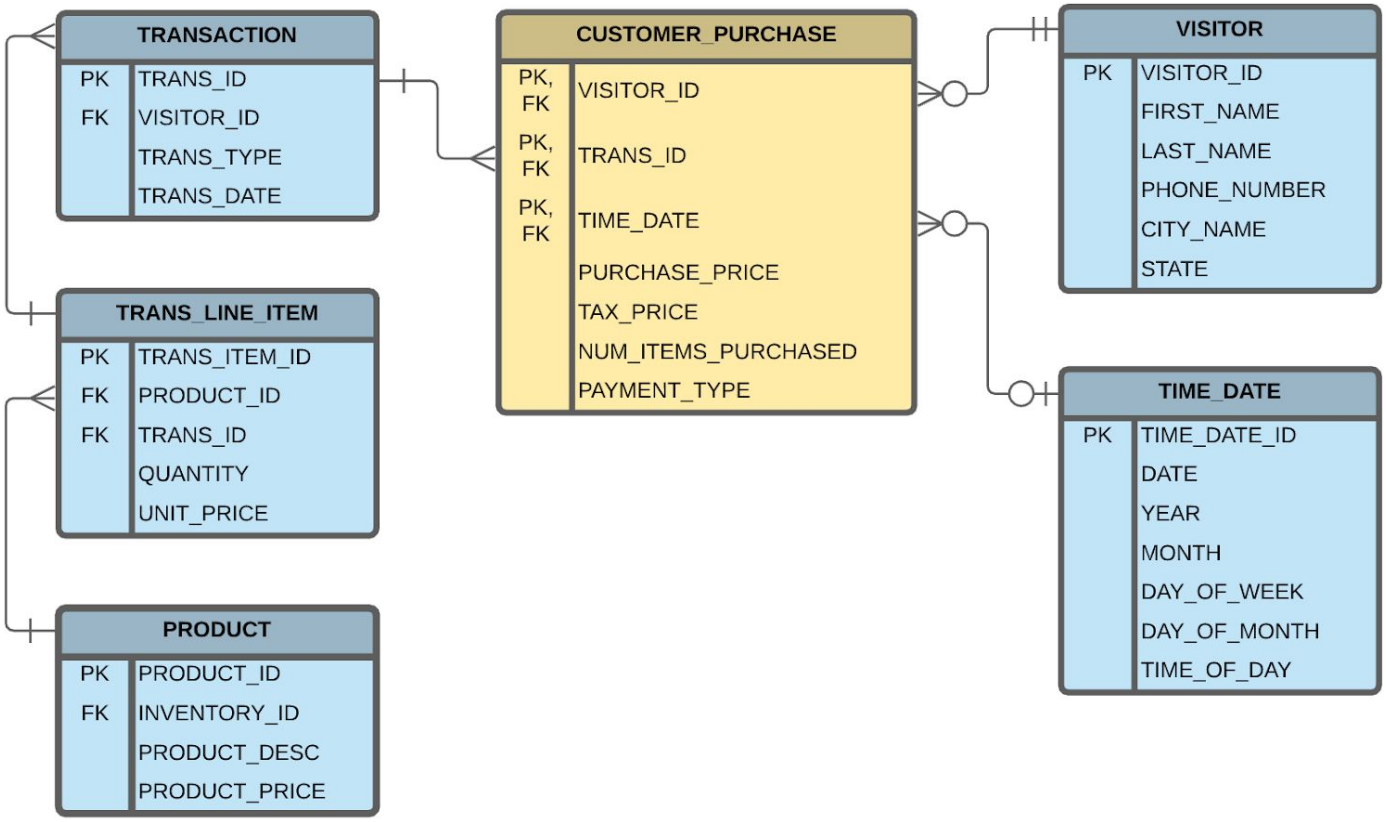
Conceptual ERD



Logical ERD


RES_LINE_ITEM
PK RES_LINE_ITEM_ID
FK ROOM_ID
FK RESERVATION_ID
ROOM
PK ROOM_ID
FK HOUSE_ID
ROOM_DESC
ROOM_RATE
VISITOR
PK VISITOR_ID
FIRST_NAME
LAST_NAME
PHONE_NUMBER
CITY_NAME
STATE
TRANSACTION
PK TRANS_ID
FK VISITOR_ID
TRANS_TYPE
TRANS_DATE
HOUSE
PK HOUSE_ID
RESERVATION
PK RESERVATION_ID
FK VISITOR_ID
CHECK_IN_DATE
CHECK_OUT_DATE
NUMBER_GUESTS
TIME_CARD
PK TIME_CARD_ID
FK EMP_ID
FK SHOP_ID
FK HOUSE_ID
FK MONK_ID
START_DATE_TIME
END_DATE_TIME
EMPLOYEE
PK EMP_ID
FIRST_NAME
LAST_NAME
PHONE_NUMBER
STREET_ADDRESS
TOWN
PAY_RATE
TRANS_LINE_ITEM
PK TRANS_ITEM_ID
FK PRODUCT_ID
FK TRANS_ID
QUANTITY
PRODUCT
PK PRODUCT_ID
FK INVENTORY_ID
PRODUCT_DESC
PRODUCT_PRICE
INVENTORY
PK INVENTORY_ID
FK SHOP_ID
QUANTITY_ON_HAND
PAYCHECK
PK PAYCHECK_ID
FK PAYROLL_ID
FK EMP_ID
CHECK_DATE
PAY_PERIOD_START
PAY_PERIOD_END
CHECK_AMOUNT
MONK
PK MONK_ID
FIRST_NAME
LAST_NAME
VOW_OF_SILENCE
PAYROLL
PK PAYROLL_ID
SHOP
PK SHOP_ID
TYPE_OF_BUSINESS

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 file.

Use Case Code

Refer to our [USE_CASE_XX.sql]() 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.

```sql
select  visitor.first_name as "visitor first name", visitor.last_name as "visitor last name", check_in_date, check_out_date,
        time_card.fk_house_id as "house id", monk.first_name as "monk first name", time_card.start_date_time from visitor
    join reservation on visitor_id = fk_visitor_id
    join res_line_item on reservation_id = fk_reservation_id
    join room on fk_room_id = room_id
    join time_card on time_card.fk_house_id = room.fk_house_id
    join monk on monk.monk_id = time_card.fk_monk_id
        where time_card.start_date_time between to_date('02-21-2021', 'mm-dd-yyyy') and to_date('02-27-2021', 'mm-dd-yyyy');
```

Query Result

SQL | All Rows Fetched: 5 in 0.106 seconds

	visitor first name	visitor last name	CHECK_IN_DATE	CHECK_OUT_DATE	house id	monk first name	START_DATE_TIME
1	JOE	SCHMIDT	01-FEB-21 00:00:00	03-FEB-21 00:00:00	h1010	Joseph	23-FEB-21 16:00:00
2	MICHAEL	SCHUMACHER	01-FEB-21 00:00:00	08-FEB-21 00:00:00	h1008	Pam	21-FEB-21 08:00:00
3	MICHAEL	SCHUMACHER	15-MAR-21 00:00:00	25-MAR-21 00:00:00	h1008	Pam	21-FEB-21 08:00:00
4	MICHAEL	SCHUMACHER	01-FEB-21 00:00:00	08-FEB-21 00:00:00	h1008	Pam	22-FEB-21 16:00:00
5	MICHAEL	SCHUMACHER	15-MAR-21 00:00:00	25-MAR-21 00:00:00	h1008	Pam	22-FEB-21 16:00:00

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…

```
7  select room_id, visitor.first_name as "visitor first name", visitor.last_name as "visitor last name" from room
8      left join res_line_item on room_id = fk_room_id
9      left join reservation on fk_reservation_id = reservation_id
10     left join visitor on fk_visitor_id = visitor_id
11         and to_date('2/02/2021', 'mm/dd/yyyy') between check_in_date and check_out_date;
```

Query Result

SQL | All Rows Fetched: 6 in 0.106 seconds

	ROOM_ID	visitor first name	visitor last name
1	1005	JOE	SCHMIDT
2	1001	GRETA	THUNBERG
3	1003	MICHAEL	SCHUMACHER
4	1004	(null)	(null)
5	1002	(null)	(null)
6	1003	(null)	(null)

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.

```
1 select visitor_id, first_name, last_name, count(*) as num_visits from reservation
2 join visitor on reservation.fk_visitor_id = visitor.visitor_id
3 group by visitor_id, first_name, last_name having count(*) > 1;
```

Script Output x | Query Result x

SQL | All Rows Fetched: 1 in 0.024 seconds

	VISITOR_ID	FIRST_NAME	LAST_NAME	NUM_VISITS
1	2021115	MICHAEL	SCHUMACHER	2

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.

```
select sum(((end_date_time - start_date_time) * 24 * pay_rate)) as pay from employee
join time_card on emp_id = fk_emp_id
where emp_id = 'EMP55555' and (start_date_time between
to_date('2/02/2021', 'mm/dd/yyyy') and to_date('03/21/2021', 'mm/dd/yyyy'));
```

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.028 seconds

	PAY
1	225

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.

```
6  select r.room_id, house_id, res.check_out_date as check_out from house h
7      join room r on h.house_id = r.fk_house_id
8      join res_line_item ln on r.room_id = ln.fk_room_id
9      join reservation res on res.reservation_id = ln.fk_reservation_id
10     where to_date('03-feb-21', 'dd-mon-yy') = res.check_out_date;
```

Query Result

SQL | All Rows Fetched: 2 in 0.036 seconds

	ROOM_ID	HOUSE_ID	CHECK_OUT
1	1001	h1001	03-FEB-21 00:00:00
2	1005	h1010	03-FEB-21 00:00:00

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:

```
1 select * from product;
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 5 in 0.034 seconds

	PRODUCT_ID	PRODUCT_DESC	PRODUCT_PRICE	FK_INVENTORY_ID
1	prod001	Bag of candy	5	item001
2	prod002	Cinnamon Candle	12	item002
3	prod003	Mini Bible	25	item003
4	prod004	Jar of Jam	7	item004
5	prod005	Collection of sermons	45	item005

```
1 select * from inventory;
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 5 in 0.03 seconds

	INVENTORY_ID	FK_SHOP_ID	QUANTITY_ON_HAND
1	item001	003	258
2	item002	002	115
3	item003	001	53
4	item004	003	113
5	item005	001	197

Our procedure:

```
8  create or replace procedure update_inventory(
9     v_product_id in varchar2,
10    v_delivery_amount in number,
11    v_price in number
12    )
13 is
14 bool_product_exists number;
15 product_id varchar2(20);
16 v_current_amount number(7);
17 v_prod_description varchar2(20) := 'Default Product Desc';
18 v_inventory_id varchar2(15);
19 v_shop_id varchar2(15);
20 bool_already_exists number(10);
21 v_random_number number(10);
22 begin
23 -- check if product exists
24   select count(*)
25   into bool_product_exists
26   from product
27   where product_id = v_product_id;
28
29   if bool_product_exists = 0
30   then
31     --dbms_output.put_line('That product does not exist Inserting');
32     select dbms_random.value(100,999) num into v_random_number FROM dual;
33     v_inventory_id := 'item' || v_random_number;
34     v_shop_id := '001';
35
36     select count(*)
37     into bool_already_exists
38     from inventory
39     where inventory_id = v_inventory_id;
40
41     WHILE bool_already_exists = 1
42     LOOP
43       select dbms_random.value(100,999) num into v_random_number FROM dual;
44       v_inventory_id := 'item' || v_random_number;
45
46       select count(*)
47       into bool_already_exists
48       from inventory
49       where inventory_id = v_inventory_id;
50     END LOOP;
51
52     insert into inventory values(v_inventory_id, v_shop_id, v_delivery_amount);
53     insert into product values(v_product_id, v_prod_description, v_price, v_inventory_id);
54   else
55     -- get shop id and inventory id
56     select shop_id, inventory_id into v_shop_id, v_inventory_id from product
57     join inventory on inventory.inventory_id = product.fk_inventory_id
58     join shop on shop.shop_id = inventory.fk_shop_id
59     where product_id = v_product_id;
60
61     --dbms_output.put_line('Product exists Updating');
62     select quantity_on_hand into v_current_amount from inventory
63     where inventory_id = v_inventory_id;
64
65     update inventory set quantity_on_hand = (v_current_amount + v_delivery_amount) where inventory_id = v_inventory_id;
66     update product set product_price = v_price where product_id = v_product_id;
67   end if;
68 end;
69 /
```

After inserting a new product:

```
1 set serveroutput on;
2 exec update_inventory('prod008', 25, 50);
```

Script Output | Task completed in 0.097 seconds

```
PL/SQL procedure successfully completed.
```

```
4 select * from product join inventory on inventory.inventory_id = product.fk_inventory_id;
```

Script Output | Query Result | SQL | All Rows Fetched: 6 in 0.026 seconds

	PRODUCT_ID	PRODUCT_DESC	PRODUCT_PRICE	FK_INVENTORY_ID	INVENTORY_ID	FK_SHOP_ID	QUANTITY_ON_HAND
1	prod001	Bag of candy	5	item001	item001	003	258
2	prod002	Cinnamon Candle	12	item002	item002	002	115
3	prod003	Mini Bible	25	item003	item003	001	53
4	prod004	Jar of Jam	7	item004	item004	003	113
5	prod005	Collection of sermons	45	item005	item005	001	197
6	prod008	Default Product Desc	50	item283	item283	001	25

After updating a product:

```
1 set serveroutput on;
2 exec update_inventory('prod008', 500, 523);
```

Script Output | Task completed in 0.251 seconds

```
PL/SQL procedure successfully completed.
```

```
4 select * from product join inventory on inventory.inventory_id = product.fk_inventory_id;
```

Script Output | Query Result | SQL | All Rows Fetched: 6 in 0.027 seconds

	PRODUCT_ID	PRODUCT_DESC	PRODUCT_PRICE	FK_INVENTORY_ID	INVENTORY_ID	FK_SHOP_ID	QUANTITY_ON_HAND
1	prod001	Bag of candy	5	item001	item001	003	258
2	prod002	Cinnamon Candle	12	item002	item002	002	115
3	prod003	Mini Bible	25	item003	item003	001	53
4	prod004	Jar of Jam	7	item004	item004	003	113
5	prod005	Collection of sermons	45	item005	item005	001	197
6	prod008	Default Product Desc	523	item283	item283	001	525

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.

```
set serveroutput on;
create or replace function get_total_number_sales
(v_shop_id varchar2, v_start_date date, v_end_date date)
return number
is
v_total_sales number;
begin
  select count(*) into v_total_sales from transaction
    join trans_line_item on trans_line_item.fk_trans_id = transaction.trans_id
    join product on product.product_id = trans_line_item.fk_product_id
    join inventory on inventory.inventory_id = product.fk_inventory_id
    join shop on shop.shop_id = inventory.fk_shop_id
    where (trans_date between v_start_date and v_end_date) and shop_id = v_shop_id;
  return v_total_sales;
end;
/
```

Script Output x | Query Result x

Task completed in 0.309 seconds

```
Function GET_TOTAL_NUMBER_SALES compiled
```

```
set serveroutput on;
select get_total_number_sales('001', to_date('14-01-2021', 'dd-mm-yyyy'), to_date('22-01-2021', 'dd-mm-yyyy'))
as total_number_of_sales from dual;
```

Script Output x | Query Result x

SQL | All Rows Fetched: 1 in 0.046 seconds

	TOTAL_NUMBER_OF_SALES
1	5

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.

```
create or replace function check_rooms_available
(v_date date)
return number
is

v_rooms_available number;

BEGIN
select count(distinct room_id) into v_rooms_available from room
    left outer join res_line_item on room.room_id = res_line_item.fk_room_id
    left outer join reservation on reservation.reservation_id = res_line_item.fk_reservation_id
    where res_line_item.fk_room_id IN
        (select distinct fk_room_id from res_line_item
            join reservation on reservation.reservation_id = res_line_item.fk_reservation_id
            where (v_date not between check_in_date and check_out_date))
            or res_line_item.fk_reservation_id is null;

return v_rooms_available;
END;
/
```

Script Output | Query Result

Task completed in 0.276 seconds

```
Function CHECK_ROOMS_AVAILABLE compiled
```

```
select check_rooms_available(to_date('01-02-2021', 'dd-mm-yyyy')) as rooms_available from dual;
```

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 1 in 0.049 seconds

	ROOMS_AVAILABLE
1	3

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:

```
-- Create the customer_bill table to store the bills.
create table customer_bill (customer_id varchar2(15) primary key not null, total_bill number);
```

Script Output | Query Result | Query Result 1

Task completed in 0.169 seconds

```
Table CUSTOMER_BILL created.
```

Function to check if the room is available:

```
-- Function to check if room is available before making a reservation
create or replace function check_if_room_available
(v_room_number varchar2, v_check_in_date date, v_check_out_date date)
return number
is
bool_is_available number;
begin
select count(*) into bool_is_available from house
    join room on house.house_id = room.fk_house_id
    join res_line_item on room.room_id = res_line_item.fk_room_id
    join reservation on res_line_item.fk_reservation_id = reservation.reservation_id
    where room.room_id = v_room_number AND (reservation.check_in_date between
    v_check_in_date and v_check_out_date)
    OR room.room_id = v_room_number AND (reservation.check_out_date between
    v_check_in_date and v_check_out_date);
    if bool_is_available = 1 then bool_is_available := 0;
    elsif bool_is_available = 0 then bool_is_available := 1;
    end if;
    return bool_is_available;
end;
/
```

Script Output | Query Result | Query Result 1

Task completed in 0.206 seconds

```
Function CHECK_IF_ROOM_AVAILABLE compiled
```

Select statement to see if the room is available:

```
-- Sample code to check if room is available
select check_if_room_available('1003', to_date('03/26/2021', 'mm/dd/yyyy'), to_date('03/30/2021', 'mm/dd/yyyy')) as room_is_available from dual;
```

Script Output | Query Result | Query Result 1 | Query Result 2

SQL | All Rows Fetched: 1 in 0.031 seconds

	ROOM_IS_AVAILABLE
1	1

Trigger on an insert into the res_line_item table:

```
-- Create the trigger on an insert into res_line_item
CREATE OR REPLACE TRIGGER visitor_total_calc
before INSERT
    ON res_line_item
FOR EACH ROW
DECLARE

    var_visitor_id varchar2(10);
    var_check_in_date date;
    var_check_out_date date;
    var_total_bill number(7,2);
    var_room_rate number(7,2);
    var_room_id varchar2(10);
    var_reservation_id varchar2(10);
    bool_room_available number;

BEGIN
  if inserting then
    select fk_visitor_id, check_in_date, check_out_date, res.reservation_id
        into var_visitor_id, var_check_in_date, var_check_out_date, var_reservation_id
        from reservation res
        where res.reservation_id = :NEW.fk_reservation_id;

        dbms_output.put_line(var_visitor_id);
        dbms_output.put_line(var_check_in_date);
        dbms_output.put_line(var_check_out_date);

        select distinct room_rate into var_room_rate from room
        join res_line_item on res_line_item.fk_room_id = room.room_id
        where room_id = :NEW.fk_room_id;

        dbms_output.put_line(var_room_rate);

        var_total_bill := (var_check_out_date - var_check_in_date) * var_room_rate;

        dbms_output.put_line(var_visitor_id  || ' is making a reservation.
        Total bill is: ' || var_total_bill);
        insert into customer_bill values(var_visitor_id, var_total_bill);
  end if;
END;
/
```

Script Output | Query Result | Query Result 1 | Query Result 2

Task completed in 0.26 seconds

```
Trigger VISITOR_TOTAL_CALC compiled
```

Empty customer_bill table:

ORCL - student17 | CUSTOMER_BILL

Columns | Data | Model | Constraints | Grants | Statistics | Triggers

Sort.. | Filter:

CUSTOME...	TOTAL_BILL

res_line_item table before:

ORCL - student17 | RES_LINE_ITEM

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback

Sort.. | Filter:

	RES_LINE_ITEM_ID	FK_ROOM_ID	FK_RESERVATION_ID
1	line3276	1001	135875
2	line3277	1003	135876
3	line3278	1004	135877
4	line3279	1005	135878
5	line3280	1003	135879

reservation table before:

ORCL - student17 | RESERVATION

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Sort.. | Filter:

	RESERVATION_ID	FK_VISITOR_ID	NUMBER_GUESTS	CHECK_IN_DATE	CHECK_OUT_DATE
1	135875	2021112	4	01-FEB-21 00:00:00	03-FEB-21 00:00:00
2	135876	2021115	8	01-FEB-21 00:00:00	08-FEB-21 00:00:00
3	135877	2021113	1	04-FEB-21 00:00:00	06-FEB-21 00:00:00
4	135878	2021111	4	01-FEB-21 00:00:00	03-FEB-21 00:00:00
5	135879	2021115	6	15-MAR-21 00:00:00	25-MAR-21 00:00:00

Insert a new reservation:

```
insert into reservation
(reservation_id, check_in_date, check_out_date, number_guests, fk_visitor_id)
values
('135554', to_date('03/26/2021', 'mm/dd/yyyy'), to_date('03/30/2021', 'mm/dd/yyyy'), 6, '2021113');
```

Script Output | Query Result | Query Result 1 | Query Result 2

Task completed in 0.153 seconds

```
1 row inserted.
```

After inserting a new res_line_item:

```
-- Insert a new res_line_item
insert into res_line_item
(res_line_item_id, fk_room_id, fk_reservation_id)
values
('line3244', '1001', '135554');
```

Script Output | Query Result | Query Result 1 | Query Re

Task completed in 0.17 seconds

```
2021113
26-MAR-21 00:00:00
30-MAR-21 00:00:00
115
2021113 is making a reservation.
        Total bill is: 460


1 row inserted.
```

customer_bill table after the insert:

ORCL - student17 | CUSTOMER_BILL

Columns | Data | Model | Constraints | Grants | Statistics

Sort.. | Filter:

	CUSTOMER_ID	TOTAL_BILL
1	2021113	460

reservation and res_line_item table after the inserts:

```
91 select * from reservation join res_line_item
92 on res_line_item.fk_reservation_id = reservation.reservation_id
93 where reservation_id = '135554';
```

Script Output | Query Result | Query Result 1 | Query Result 2

SQL | All Rows Fetched: 1 in 0.029 seconds

	RESERVATION_ID	FK_VISITOR_ID	NUMBER_GUESTS	CHECK_IN_DATE	CHECK_OUT_DATE	RES_LINE_ITEM_ID	FK_ROOM_ID	FK_RESERVATION_ID
1	135554	2021113	6	26-MAR-21 00:00:00	30-MAR-21 00:00:00	line3244	1001	135554

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:

ORCL - student17 | INVENTORY

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashb

Sort.. | Filter:

	INVENTORY_ID	FK_SHOP_ID	QUANTITY_ON_HAND
1	item001	003	258
2	item002	002	115
3	item003	001	53
4	item004	003	113
5	item005	001	197
6	item283	001	525

Package spec:

```
CREATE OR REPLACE PACKAGE transaction_calc_package AS

    CURSOR transaction_information(p_trans_id varchar2) IS
        select * from transaction
            join trans_line_item on trans_line_item.fk_trans_id = transaction.trans_id
            join visitor on visitor.visitor_id = transaction.fk_visitor_id
            join product on product.product_id = trans_line_item.fk_product_id
            join inventory on inventory.inventory_id = product.fk_inventory_id
            where transaction.trans_id = p_trans_id;

    FUNCTION sales_tax_calc(p_visitor_id varchar2, p_trans_id varchar2) return number;

    PROCEDURE handle_transaction_proc (p_visitor_id IN varchar2, p_trans_id IN varchar2);

END transaction_calc_package;
/
```

Package body:

```
CREATE OR REPLACE PACKAGE BODY transaction_calc_package AS
    FUNCTION sales_tax_calc(
        p_visitor_id in varchar2,
        p_trans_id in varchar2)
        RETURN number
    IS
        var_visitor_state varchar2(25);
        var_total_paid number(7,2) := 0;
        var_sales_tax_amount number(7,2);
    BEGIN
        FOR rec IN transaction_information(p_trans_id)
        LOOP
                var_total_paid := var_total_paid + (rec.quantity * rec.product_price);
                var_visitor_state := lower(rec.state);
        END LOOP;
        case
            when var_visitor_state = 'wa' then var_sales_tax_amount := var_total_paid * 0.05;
            when var_visitor_state = 'or' then var_sales_tax_amount := var_total_paid * 0.08;
            when var_visitor_state = 'ca' then var_sales_tax_amount := var_total_paid * 0.1;
            when var_visitor_state = 'switzerland' then var_sales_tax_amount := var_total_paid * 0.15;
            when var_visitor_state = 'sweden' then var_sales_tax_amount := var_total_paid * 0.20;
        end case;
        dbms_output.put_line('Total paid: ' || var_total_paid);
        dbms_output.put_line('Total tax: ' || var_sales_tax_amount);
        RETURN var_sales_tax_amount;
    END sales_tax_calc;
    PROCEDURE handle_transaction_proc(
        p_visitor_id IN varchar2,
        p_trans_id IN varchar2)
    IS
        var_visitor_name varchar2(50);
        var_visitor_state varchar2(25);
        var_total_paid number(7,2) := 0;
        var_total_sales_tax number(7,2);
        var_total_with_tax number(7,2);
        var_transaction_date date;
    BEGIN
        FOR rec IN transaction_information(p_trans_id)
        LOOP
            var_total_paid := var_total_paid + (rec.quantity * rec.product_price);
            var_visitor_state := lower(rec.state);
            var_visitor_name := rec.first_name || ' ' || rec.last_name;
            var_transaction_date := rec.trans_date;
            -- remove from inventory and update
            update inventory i set i.quantity_on_hand = (i.quantity_on_hand - rec.quantity)
            where i.inventory_id = rec.fk_inventory_id;
        END LOOP;
        var_total_sales_tax := sales_tax_calc(p_visitor_id, p_trans_id);
        var_total_with_tax := var_total_paid + var_total_sales_tax;
        dbms_output.put_line('Visitor ID: ' || p_visitor_id);
        dbms_output.put_line('Visitor name: ' || var_visitor_name);
        dbms_output.put_line('Transaction date: ' || var_transaction_date);
        dbms_output.put_line('Total amount paid(with tax): ' || var_total_with_tax);
    END handle_transaction_proc;
END transaction_calc_package;
/
```

Running the package:

```
-- Sample code to test the package
SET SERVEROUTPUT ON;
DECLARE
    visitor_id varchar2(10) := '2021112';
    transaction_id varchar2(10) := '2021002';
    v_sales_tax number(7,2);
BEGIN
    dbms_output.put_line('Calculate Sales Tax');
    dbms_output.put_line('--------------------');
    v_sales_tax := transaction_calc_package.sales_tax_calc(visitor_id, transaction_id);
    dbms_output.put_line('--------------------');
    transaction_calc_package.handle_transaction_proc(visitor_id, transaction_id);
END;
/
```

Script Output

Task completed in 0.229 seconds

```
Calculate Sales Tax
--------------------
Total paid: 228
Total tax: 45.6
--------------------
Total paid: 228
Total tax: 45.6
Visitor ID: 2021112
Visitor name: GRETA THUNBERG
Transaction date: 16-JAN-21 00:00:00
Total amount paid(with tax): 273.6


PL/SQL procedure successfully completed.
```

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

ORCL - student17 | INVENTORY

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashb

Sort.. | Filter:

	INVENTORY_ID	FK_SHOP_ID	QUANTITY_ON_HAND
1	item001	003	258
2	item002	002	111
3	item003	001	53
4	item004	003	113
5	item005	001	193
6	item283	001	525

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);

2. 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);