# Project – Database Foundation

Project Title: DBF 12

Tool – MySQL Workbench

DBF 12 - Technical Report

**Table of Contents**

1. Introduction
2. Requirements Definition Document
   1. Entity and Attribute Description
   2. Relationship, Cardinality Description and Business Rules
   3. Assumptions and Special Considerations
3. Detailed Database Design – Entity Relationship Diagram (ERD)
4. DDL Source Code
5. DML Source Code
6. Query Source Code
7. DDL Output
8. DML Output
9. Query Output

# Introduction

In this Database Design Document, we will see the design and implementation of a database for the front desk of our new hotel that has a theme park on the hotel premises itself. The Hotel is named as DBF 12 in loving memory of our database foundations class taught by Prof. Gasan Elkhodari. The Number 12 is derived from Group 12 that we are a part of and was generated by an auto-generator. As this document progresses, we will see the design and implementation of the database for the front desk of hotel DBF 12 that will store all the guest information with details, the rooms and the amenities they offer including the rates for it, reservation management system, pricing policies and finally computing total payment for our guests based on their selections, any discounts offered to them and storing the payment details. So, check out our new hotel DBF 12 that has its own theme park for your next vacation!

# Requirements Definition Document

* 1. **Entity and Attribute Description**

Entity Name: Guest

Entity Description: Guest is the primary entity around whom all the other entities would revolve. It includes details on all guests.

Main Attributes of Guest:

**Guest\_ID:** (Primary Key) A unique identifier of the guest’s account in Hotel DBF 12’s database.

L\_Name: Last Name of the guest who is the primary guest under whose name booking is made and who is responsible for clearing payment upon checkout

F\_Name: First Name of the guest who is the primary guest under whose name booking is made and who is responsible for clearing payment upon checkout

Age: Age of the primary guest under whose name booking is made and who is responsible for clearing payment upon checkout. It is important to know the age as the hotel requires an adult to check-in. Also, the hotel offers a discount to senior citizens who are of the age 65 years and above.

Email\_ID: Email\_ID of the guest who is the primary guest under whose name booking is made and who is responsible for clearing payment upon checkout. Email\_ID is used for the purposes of sending reservation confirmation, discount coupons and a copy of payment receipt.

Phone: It identifies the 10 digit phone number of the guest who is the primary guest under whose name booking is made and who is responsible for clearing payment upon checkout Platinum\_Member: Identifier of whether a guest has a platinum membership at Hotel DBF 12 or not. It is identified as ‘Yes’ if the guest is a platinum member of Hotel DBF 12 and ‘No’ if the guest is not a platinum member of Hotel DBF 12.

Entity Name: Reservation

Entity Description: Reservation Entity reflects the reservation details of the guest like their planned stay at the hotel, purchase of theme park tickets and how far in advanced they made the reservation.

Main Attributes of Reservation:

**Res\_No:** (Primary Key) A unique identifier of the guest’s reservation in Hotel DBF 12’s database.

Guest\_ID: (Foreign Key) References the Guest table and connects each unique reservation to one guest.

Checkin\_Dt: Date when the guest checks in at Hotel DBF 12. Checkout\_Dt: Date when the guest checks out of Hotel DBF 12.

No\_of\_Nights: The total stay a guest is at the hotel for. It is computed through subtracting the Checkin\_Dt from the Checkout\_Dt

Res\_Status: It reflects the status of the guest’s reservation. Whether the reservation is confirmed or is it in waiting status for a room to become available for the dates requested.

No\_of\_Guests : This is the total number of guests on a particular reservation that includes the primary guest who checked-in and all the other guests accompanying him/her.

No\_of\_TP\_Tickets: It gives the count of DBF 12 theme park tickets a guest purchased along with their room reservation.

Res\_Date: Date when the guest made the reservation. It is an important factor in deciding the discount for the guest. If the guest made the reservation 30 days before their check-in date, they would get an early bird discount of 10%.

Entity Name: Rooms

Entity Description: It describes the various features of a room that is reserved by a guest. Main Attributes of Rooms:

**Room\_No:** (Primary Key) A unique identifier of each room. It is not auto generated but manually assigned by the Hotel Staff based on the attributes and amenities a customer selects while making reservations.

Res\_No:(Foreign Key) References the Reservation table and connects each unique reservation to the room reserved under that reservation.

Room\_Category: It categorizes a room into either Standard King, Standard Queen, Standard Twin or an Executive Suite.

Room\_Type: It indicates whether a room is a smoking room or a non-smoking room Room\_View: It indicates whether a room has a Pool View, Street View, Themepark View or a Garden View

Breakfast: It is a ‘Yes’ or ‘No’ indicator which indicates whether a buffet breakfast is included with the room reservation or not.

Room\_Rate: It is a rate predetermined by the owner of the Hotel DBF 12 who has provided a specific non-changeable room rate card to the hotel front desk staff that is to be used for making any reservation and collection of final payment.

Entity Name: Pricing

Entity Description: Pricing table plays a crucial role in determining the final payment by indicating how much to charge each customer for their hotel stay and theme park tickets and what discounts to apply.

Main Attributes of Pricing:

**Pricing\_ID:** (Primary Key) A unique identifier that indicates what price to charge based on the discounts a guest is eligible for. It follows the DBF 12 clause that only one type of discount can be offered to a guest even if they are eligible for more than one type of discount. The discount

with the maximum percentage off will be offered to the guest and the Pricing ID associated with that discount will be used to indicate the payment they should be charged.

Discount\_Code: A unique identifier that identifies the discount offered to a guest. Discount\_Type: It indicates the type of discount offered to a guest based on their eligibility. Discount\_percent: It indicates the percentage off a guest would receive on their total bill based on their eligibility.

Theme\_Park\_Price: Kids under the age of 6 get a free theme park entry. A theme park entry ticket is required for anyone age of 6 and above who wishes to enjoy theme park rides and is sold at a flat rate of $55. A hotel guest is however not required to purchase the theme park ticket. It is their choice if they purchase the theme park ticket along with their stay at the hotel. If they purchase more than two theme park tickets, they get a discount of 5%. If they reserve a room and purchase a theme park ticket they get a discount of 20%. Again, as per DBF 12 clause, if a guest is eligible for more than one discount, only one discount would be applicable, whichever is higher.

Room\_Tax\_Rate: It indicates the total of city and state tax rate on hotel rooms which is 13% for Richardson where Hotel DBF 12 is located at. The only ones exempt from Room\_Tax\_Rate are guests with a state of Texas tax exempt certificate. In that event, the front desk would use the Pricing ID of 6 to indicate that they are Tax Exempt.

TP\_Ticket\_Tax\_Rate: It indicates the sales tax of 8.25% to be charged on the purchase of each theme park ticket. The only ones exempt from TP\_Ticket\_Tax\_Rate are guests with a state of Texas tax exempt certificate. In that event, the front desk would use the Pricing ID of 6 to indicate that they are Tax Exempt.

Entity Name: Payment

Entity Description: The Payment entity indicates the payment amount due for a guest for their stay at the hotel and purchase of theme park tickets and the payment details.

Main Attributes of Payment:

**Pmt ID:** (Primary Key) : A unique identifier that indicates the Payment ID associated with a particular reservation for a guest.

Res\_No (Foreign Key): It references to the Reservation table and indicates the choices of the guest made under this reservation, the length of their stay and the number of theme park tickets purchased by them(if applicable).

Pricing\_ID(Foreign Key): It references to the Pricing table. The Pricing ID is determined, based on the discount a guest is eligible for. It follows the DBF 12 clause that only one type of discount can be offered to a guest even if they are eligible for more than one type of discount. The discount with the maximum percentage off will be offered to the guest and the Pricing ID associated with that discount will be used to indicate the payment they should be charged.

Pmt\_Method: It would indicate whether a guest would be using a credit card or a debit card for the payment of their hotel room stay and theme park ticket purchase if applicable.

Card\_Number: A 16 digit number unique to each credit or debit card used to make the payment. Type\_of\_Card: It would indicate whether the card is a Visa or a Mastercard.

Name\_on\_Card: This would indicate the First name and Last name of the guest generated by concatenating the First Name and Last Name from the Guest Table

Exp\_Date: This would indicate the expiration date of the credit or debit card used to make the payment by the guest.

Pmt\_Status: This would indicate whether a guest who made a reservation at the hotel has paid in full and cleared their invoice or do they have a balance due. The balance due amount will be listed under the Final Payment Amount. The Final Payment Amount is subject to change based on the guest’s addition to additional theme park tickets or their extending the stay at the hotel.

Hence, the Final Payment Amount is calculated for the guest during the check-out process.

* 1. **Relationship, Cardinality Description and Business Rules** Relationship: has between GUEST and RESERVATION. Cardinality: 1:M between GUEST and RESERVATION.

Business rule: A GUEST may have zero, one or two RESERVATION at a given point in time; One RESERVATION must be associated with only one GUEST.

Relationship: shows between RESERVATION and ROOMS. Cardinality: 1:1 between RESERVATION and ROOMS.

Business rule: Each RESERVATION shows details on one ROOM associated with it. Each ROOM is associated with only one RESERVATION. Thus, only one ROOM can be booked per RESERVATION identified by a unique Reservation Number. If multiple ROOMS need to be booked, multiple RESERVATION under different Reservation Number should be made by the GUEST.

Relationship: associates between RESERVATION and PAYMENT. Cardinality: 1:1 between RESERVATION and PAYMENT.

Business rule: Each RESERVATION is associated with a unique PAYMENT. Each PAYMENT is associated with only one RESERVATION.

Relationship: determines between PRICING and PAYMENT. Cardinality: 1:1 between PRICING and PAYMENT.

Business rule: Each PRICING ID determines a unique PAYMENT. Each PAYMENT is associated with only one PRICING ID. This is possible through the hotel DBF 12 clause that only one Discount Code can be used at a time. If a guest is eligible for more than one type of Discount Code, the one with the higher percentage will be considered. Thus, their PAYMENT with only be determined by one PRICING ID that is associated with the higher discount code.

# BUSINESS RULES:

1. A GUEST may have zero, one or two RESERVATIONS at a given point in time
2. Each RESERVATION must be associated with only one GUEST.
3. Each RESERVATION shows details on one ROOM associated with it.
4. Each ROOM is associated with only one RESERVATION.
5. Each RESERVATION is associated with a unique PAYMENT.
6. Each PAYMENT is associated with only one RESERVATION.
7. Each PRICING ID determines a unique PAYMENT.
8. Each PAYMENT is associated with only one PRICING ID.

# Assumptions and Special Considerations

* + 1. Hotel DBF 12 clause is that only one Discount Code can be used at a time. If a guest is eligible for more than one type of Discount Code, the one with the higher percentage will be

considered. Thus, their PAYMENT will only be determined by one PRICING ID that is associated with the higher discount code.

* + 1. Kids under the age of 6 get a free theme park entry. A theme park entry ticket is required for anyone age of 6 and above and is sold at a flat rate of $55.
    2. Platinum Membership at DBF 12 is earned by accumulating points. When a Guest accumulates 1000 points, they are awarded platinum membership. Each stay at Hotel DBF 12 leads to accumulation of 100 points. Once a guest is a platinum member, they are eligible for the Platinum member discount of 15% and access to the Hotel’s VIP lounge on the Roof top. The Platinum Membership points and further details are maintained by our marketing staff in their database. The Guest table in the front desk database would only have an indication whether they are a platinum member or not.
    3. Guest records remain on DBF 12’s database for 3 years. After that they are purged. So, a guest who has been to the hotel in last three years will show up even if they don’t have a current reservation at the hotel. Also, DBF 12 accepts walk-ins, so we will let a guest check-in instantly without a prior reservation on record. Upon check-in, they will be assigned an auto-generated sequential Res No that will be associated with their account.
    4. Hotel DBF 12’s owner is very particular and has pre-determined the nightly rate for each room based on the Room Category, Type of Room, Type of View it Offers, whether it is a Smoking or Non-Smoking Room and whether a guest chooses to include Buffet Breakfast or Not as a part of their reservation. The Front Desk is expected to enter this amount for the Room Rate. The Owner has shared her Room Rate Card below:

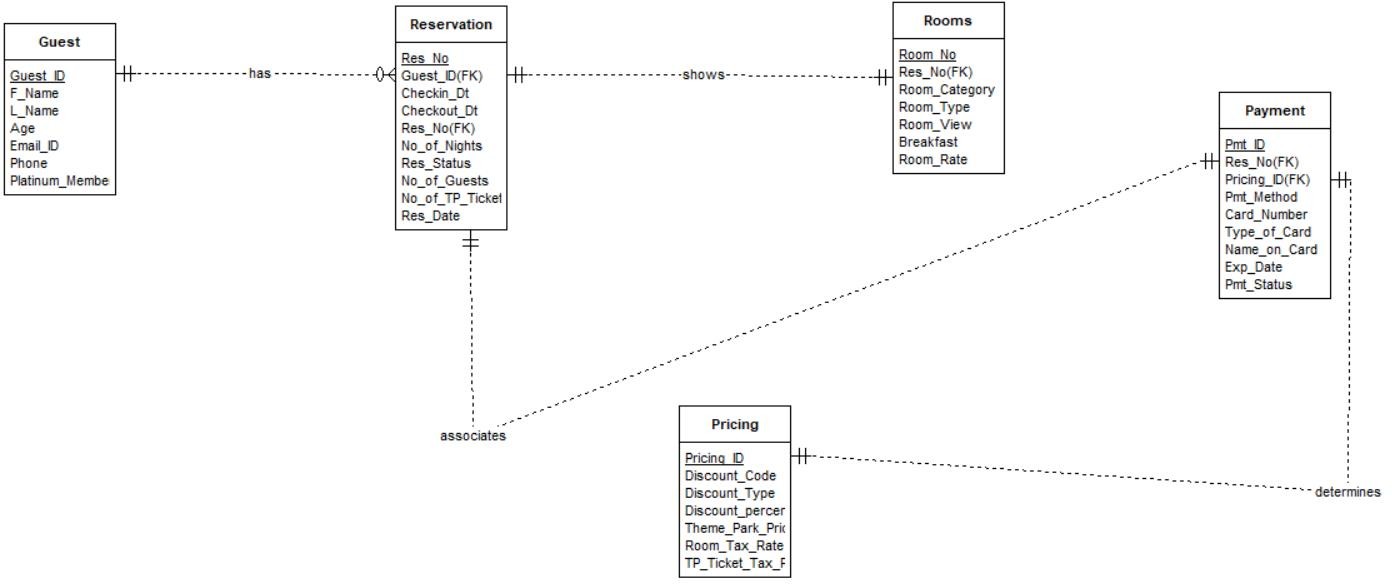
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Category | View | Type | Breakfast | Room Rate/night |
| Standard King | Street View | Non- Smoking | Yes | $170 |
| Standard King | Street View | Non- Smoking | No | $160 |
| Standard King | Street View | Smoking | Yes | $175 |
| Standard King | Street View | Smoking | No | $165 |
| Standard King | Themepark View | Non- Smoking | Yes | $185 |
| Standard King | Themepark View | Non- Smoking | No | $175 |
| Standard King | Themepark View | Smoking | Yes | $190 |
| Standard King | Themepark View | Smoking | No | $180 |
| Standard King | Pool View | Non- Smoking | Yes | $180 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Standard King | Pool View | Non- Smoking | No | $170 |
| Standard King | Pool View | Smoking | Yes | $185 |
| Standard King | Pool View | Smoking | No | $175 |
| Standard King | Garden View | Non- Smoking | Yes | $175 |
| Standard King | Garden View | Non- Smoking | No | $165 |
| Standard King | Garden View | Smoking | Yes | $180 |
| Standard King | Garden View | Smoking | No | $170 |
| Standard Queen | Street View | Non- Smoking | Yes | $150 |
| Standard Queen | Street View | Non- Smoking | No | $140 |
| Standard Queen | Street View | Smoking | Yes | $155 |
| Standard Queen | Street View | Smoking | No | $145 |
| Standard Queen | Themepark View | Non- Smoking | Yes | $165 |
| Standard Queen | Themepark View | Non- Smoking | No | $155 |
| Standard Queen | Themepark View | Smoking | Yes | $170 |
| Standard Queen | Themepark View | Smoking | No | $160 |
| Standard Queen | Pool View | Non- Smoking | Yes | $160 |
| Standard Queen | Pool View | Non- Smoking | No | $150 |
| Standard Queen | Pool View | Smoking | Yes | $165 |
| Standard Queen | Pool View | Smoking | No | $155 |
| Standard Queen | Garden View | Non- Smoking | Yes | $155 |
| Standard Queen | Garden View | Non- Smoking | No | $145 |
| Standard Queen | Garden View | Smoking | Yes | $160 |
| Standard Queen | Garden View | Smoking | No | $150 |
| Standard Twin | Street View | Non- Smoking | Yes | $130 |
| Standard Twin | Street View | Non- Smoking | No | $120 |
| Standard Twin | Street View | Smoking | Yes | $135 |
| Standard Twin | Street View | Smoking | No | $125 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Standard Twin | Themepark View | Non- Smoking | Yes | $145 |
| Standard Twin | Themepark View | Non- Smoking | No | $135 |
| Standard Twin | Themepark View | Smoking | Yes | $150 |
| Standard Twin | Themepark View | Smoking | No | $140 |
| Standard Twin | Pool View | Non- Smoking | Yes | $140 |
| Standard Twin | Pool View | Non- Smoking | No | $130 |
| Standard Twin | Pool View | Smoking | Yes | $145 |
| Standard Twin | Pool View | Smoking | No | $135 |
| Standard Twin | Garden View | Non- Smoking | Yes | $135 |
| Standard Twin | Garden View | Non- Smoking | No | $125 |
| Standard Twin | Garden View | Smoking | Yes | $140 |
| Standard Twin | Garden View | Smoking | No | $130 |
| Executive Suite | Street View | Non- Smoking | Yes | $210 |
| Executive Suite | Street View | Non- Smoking | No | $200 |
| Executive Suite | Street View | Smoking | Yes | $215 |
| Executive Suite | Street View | Smoking | No | $205 |
| Executive Suite | Themepark View | Non- Smoking | Yes | $225 |
| Executive Suite | Themepark View | Non- Smoking | No | $215 |
| Executive Suite | Themepark View | Smoking | Yes | $230 |
| Executive Suite | Themepark View | Smoking | No | $220 |
| Executive Suite | Pool View | Non- Smoking | Yes | $220 |
| Executive Suite | Pool View | Non- Smoking | No | $210 |
| Executive Suite | Pool View | Smoking | Yes | $225 |
| Executive Suite | Pool View | Smoking | No | $215 |
| Executive Suite | Garden View | Non- Smoking | Yes | $215 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Executive Suite | Garden View | Non- Smoking | No | $205 |
| Executive Suite | Garden View | Smoking | Yes | $220 |
| Executive Suite | Garden View | Smoking | No | $210 |

# Detailed Database Design – Entity Relationship Diagram (ERD)



1. **DDL Source Code**

/\*Project DBF 12- DDL\*/

/\* Create tables based on entities \*/

CREATE TABLE Guest

(

Guest\_ID INTEGER AUTO\_INCREMENT NOT NULL,

F\_Name VARCHAR(25) NOT NULL,

L\_Name VARCHAR(25) NOT NULL,

AGE INTEGER NOT NULL,

Email\_ID VARCHAR(40) NOT NULL,

Phone VARCHAR(20) NOT NULL,

Platinum\_Member VARCHAR(3) NOT NULL,

CONSTRAINT PK\_Guest PRIMARY KEY (Guest\_ID)

)AUTO\_INCREMENT = 101;

CREATE TABLE Reservation

(

Res\_no INTEGER AUTO\_INCREMENT NOT NULL,

Guest\_ID INTEGER NOT NULL,

Checkin\_Dt DATE NOT NULL,

Checkout\_Dt DATE NOT NULL,

No\_of\_Nights INTEGER GENERATED ALWAYS AS (DATEDIFF(Checkout\_Dt, Checkin\_Dt)) STORED,

Res\_Status VARCHAR(50) NOT NULL,

No\_of\_Guests INTEGER NOT NULL,

No\_of\_TP\_Tickets INTEGER NOT NULL,

Res\_Date DATE NOT NULL,

CONSTRAINT PK\_Reservation PRIMARY KEY (Res\_no),

CONSTRAINT FK\_Reservation FOREIGN KEY (Guest\_ID) REFERENCES Guest(Guest\_ID)

) AUTO\_INCREMENT = 10025;

Create TABLE Rooms

(

Room\_No INTEGER NOT NULL,

Res\_No INTEGER NOT NULL,

Room\_Category VARCHAR(30) NOT NULL,

Room\_Type VARCHAR(30) NOT NULL,

Room\_View VARCHAR(30) NOT NULL,

Breakfast VARCHAR(5) NOT NULL,

Room\_Rate VARCHAR(15) NOT NULL,

CONSTRAINT PK\_Rooms PRIMARY KEY (Room\_No),

CONSTRAINT FK\_Rooms FOREIGN KEY (Res\_No) REFERENCES Reservation (Res\_No)

);

CREATE TABLE Pricing

(

Pricing\_ID INTEGER AUTO\_INCREMENT NOT NULL,

Discount\_Code VARCHAR(4) NOT NULL,

Discount\_Type VARCHAR(35) NOT NULL,

Discount\_percent INTEGER NOT NULL,

Theme\_Park\_Price INTEGER NOT NULL,

Room\_Tax\_Rate INTEGER NOT NULL,

TP\_Ticket\_Tax\_Rate INTEGER NOT NULL,

CONSTRAINT PK\_Pricing PRIMARY KEY (Pricing\_ID)

)AUTO\_INCREMENT = 1;

CREATE TABLE Payment

(

Pmt\_ID INTEGER AUTO\_INCREMENT NOT NULL,

Res\_No INTEGER NOT NULL,

Pricing\_ID INTEGER NOT NULL,

Pmt\_Method VARCHAR(35) NOT NULL,

Card\_Number VARCHAR(16) NOT NULL,

Type\_of\_Card VARCHAR(16) NOT NULL,

Name\_on\_Card VARCHAR(50) NOT NULL,

Exp\_Date VARCHAR(7) NOT NULL,

Pmt\_Status VARCHAR(20) NOT NULL,

CONSTRAINT PK\_Payment PRIMARY KEY (Pmt\_ID),

CONSTRAINT FK\_Payment FOREIGN KEY (Res\_No) REFERENCES Reservation (Res\_No),

CONSTRAINT FK2\_Payment FOREIGN KEY (Pricing\_ID) REFERENCES Pricing (Pricing\_ID)

)AUTO\_INCREMENT = 130045;

/\* Create indices for natural keys, foreign keys, and frequently-queried columns \*/

-- Guest

-- Natural keys

CREATE INDEX IDX\_Guest\_FName ON Guest (F\_Name);

-- Frequently-queried columns

CREATE INDEX IDX\_Guest\_Age ON Guest (Age);

CREATE INDEX IDX\_Guest\_Platinum\_Membership ON Guest (Platinum\_Member);

-- Reservation

-- Frequently-queried columns

CREATE INDEX IDX\_Reservation\_No\_of\_Nights ON Reservation (No\_of\_Nights);

CREATE INDEX IDX\_Reservation\_No\_of\_TP\_Tickets ON Reservation (No\_of\_TP\_Tickets);

-- Foreign Key

CREATE INDEX IDX\_Guest\_ID ON Reservation (Guest\_ID);

-- Rooms

-- Frequently-queried columns

CREATE INDEX IDX\_Room\_View ON Rooms (Room\_View);

CREATE INDEX IDX\_Room\_Rate ON Rooms (Room\_Rate);

-- Pricing

-- Frequently-queried columns

CREATE INDEX IDX\_Pricing\_Discount ON Pricing (Discount\_percent);

CREATE INDEX IDX\_Pricing\_Discount\_Type ON Pricing (Discount\_Type);

-- Payment

-- Natural keys

CREATE INDEX IDX\_Payment\_Name\_on\_Card ON Payment (Name\_on\_Card);

-- Frequently-queried columns

CREATE INDEX IDX\_Payment\_Card\_number ON Payment (Card\_Number);

-- Foreign Keys

CREATE INDEX IDX\_Payment\_Res\_No ON Payment (Res\_No);

CREATE INDEX IDX\_Paymnet\_Pricing\_ID ON Payment (Pricing\_ID);

/\* Alter Tables by adding Audit Columns \*/

ALTER TABLE Guest

ADD (

created\_by VARCHAR(30), date\_created DATE, modified\_by VARCHAR(30), date\_modified DATE

);

ALTER TABLE Reservation

ADD (

created\_by VARCHAR(30), date\_created DATE, modified\_by VARCHAR(30), date\_modified DATE

);

ALTER TABLE Rooms

ADD (

created\_by VARCHAR(30), date\_created DATE, modified\_by VARCHAR(30), date\_modified DATE

);

ALTER TABLE Pricing

ADD (

created\_by VARCHAR(30), date\_created DATE, modified\_by VARCHAR(30), date\_modified DATE

);

ALTER TABLE Payment

ADD (

created\_by VARCHAR(30), date\_created DATE, modified\_by VARCHAR(30), date\_modified DATE

);

/\* Create Views \*/

-- Business purpose: The GuestInfo view will be used primarily for quickly searching information about individual guests for populating their Age and whether they are a Platinum member or not for computing discounts.

CREATE OR REPLACE VIEW GuestInfo AS

SELECT Guest\_ID, F\_Name, L\_Name, Age, Platinum\_Member

FROM Guest;

-- Business purpose: The ReservationInfo view will be used primarily for quickly searching information about individual guests for populating the number of theme park tickets they purchased and the number of nights they would be staying at the hotel to compute the payment amount for them

CREATE OR REPLACE VIEW ReservationInfo AS

SELECT Res\_no, Guest\_ID, No\_of\_Nights,No\_of\_TP\_Tickets

FROM Reservation;

-- Business purpose: The RoomsInfo view will be used primarily for quickly searching about individual guests for populating the room type and room view and the price of the room

CREATE OR REPLACE VIEW RoomsInfo AS

SELECT Room\_No, Room\_Category, Room\_Type, Room\_View, Room\_Rate

FROM Rooms;

-- Business purpose: The PricingInfo view will be used primarily for quickly searching information about the Discount Code and Discount Type to derive the Pricing\_ID.

CREATE OR REPLACE VIEW PricingInfo AS

SELECT Pricing\_ID, Discount\_Type, Discount\_percent

FROM Pricing;

-- Business purpose: The PaymentInfo view will be used primarily for quickly searching information about the name of card holder and respective Pmt\_ID and Res\_No associated with it.

CREATE OR REPLACE VIEW PaymentInfo AS

SELECT Pmt\_ID, Res\_No, Name\_on\_Card

FROM Payment;

/\* Create Triggers \*/

-- Business purpose: The TR\_Guest trigger automatically assigns a sequential Guest ID to a newly-inserted row in the Guest table, assigning appropriate values to the created\_by and date\_created fields. If the record is being inserted or updated, appropriate values are assigned to the modified\_by and modified\_date fields.

DELIMITER //

CREATE TRIGGER TR\_Guest BEFORE INSERT ON Guest

FOR EACH ROW

BEGIN

IF NEW.Guest\_ID IS NULL THEN

SET NEW.Guest\_ID = (SELECT AUTO\_INCREMENT FROM information\_schema.TABLES WHERE TABLE\_NAME = 'Guest' AND TABLE\_SCHEMA = DATABASE()) + 1;

END IF;

IF NEW.created\_by IS NULL THEN

SET NEW.created\_by = USER();

END IF;

IF NEW.date\_created IS NULL THEN

SET NEW.date\_created = NOW();

END IF;

SET NEW.modified\_by = USER();

SET NEW.date\_modified = NOW();

END;

//

DELIMITER ;

-- Business purpose: The TR\_Res trigger automatically assigns a sequential Reservation Number to a newly-inserted row in the Reservation table, assigning appropriate values to the created\_by and date\_created fields. If the record is being inserted or updated, appropriate values are assigned to the modified\_by and modified\_date fields.

DELIMITER //

CREATE TRIGGER TR\_Res

BEFORE INSERT ON Reservation FOR EACH ROW

BEGIN

IF NEW.Res\_No IS NULL THEN

SET NEW.Res\_No = (SELECT NEXTVAL('SEQ\_Res\_No'));

END IF;

IF NEW.created\_by IS NULL THEN

SET NEW.created\_by = CURRENT\_USER();

END IF;

IF NEW.date\_created IS NULL THEN

SET NEW.date\_created = NOW();

END IF;

SET NEW.modified\_by = CURRENT\_USER();

SET NEW.date\_modified = NOW();

END;

//

DELIMITER ;

-- Business purpose: The TR\_Rooms trigger automatically assigns appropriate values to the created\_by and date\_created fields. If the record is being inserted or updated, appropriate values are assigned to the modified\_by and modified\_date fields.

DELIMITER //

CREATE TRIGGER TR\_Rooms

BEFORE INSERT ON Rooms FOR EACH ROW

BEGIN

IF NEW.created\_by IS NULL THEN

SET NEW.created\_by = CURRENT\_USER();

END IF;

IF NEW.date\_created IS NULL THEN

SET NEW.date\_created = NOW();

END IF;

SET NEW.modified\_by = CURRENT\_USER();

SET NEW.date\_modified = NOW();

END;

//

DELIMITER ;

-- Business purpose: The TR\_Pricing trigger automatically assigns a sequential Pricing ID to a newly-inserted row in the Pricing table, assigning appropriate values to the created\_by and date\_created fields. If the record is being inserted or updated, appropriate values are assigned to the modified\_by and modified\_date fields.

DELIMITER //

CREATE TRIGGER TR\_Pricing

BEFORE INSERT ON Pricing FOR EACH ROW

BEGIN

IF NEW.Pricing\_ID IS NULL THEN

SET NEW.Pricing\_ID = (SELECT NEXTVAL('SEQ\_Pricing\_ID'));

END IF;

IF NEW.created\_by IS NULL THEN

SET NEW.created\_by = CURRENT\_USER();

END IF;

IF NEW.date\_created IS NULL THEN

SET NEW.date\_created = NOW();

END IF;

SET NEW.modified\_by = CURRENT\_USER();

SET NEW.date\_modified = NOW();

END;

//

DELIMITER ;

-- Business purpose: The TR\_Payment trigger automatically assigns a sequential Pmt ID to a newly-inserted row in the Payment table, assigning appropriate values to the created\_by and date\_created fields. If the record is being inserted or updated, appropriate values are assigned to the modified\_by and modified\_date fields.

DELIMITER //

CREATE TRIGGER TR\_Payment

BEFORE INSERT ON Payment FOR EACH ROW

BEGIN

IF NEW.Pmt\_ID IS NULL THEN

SET NEW.Pmt\_ID = (SELECT NEXTVAL('SEQ\_Pmt\_ID'));

END IF;

IF NEW.created\_by IS NULL THEN

SET NEW.created\_by = CURRENT\_USER();

END IF;

IF NEW.date\_created IS NULL THEN

SET NEW.date\_created = NOW();

END IF;

SET NEW.modified\_by = CURRENT\_USER();

SET NEW.date\_modified = NOW();

END;

//

DELIMITER ;

/\* Populate all tables \*/

-- Guest table

INSERT INTO Guest (F\_Name, L\_Name, Age, Email\_ID, Phone, Platinum\_Member)

VALUES ('Gasan', 'Elkhodari', 42, 'gasan.elkhodari@utdallas.edu', '(972)883-4779', 'Yes');

INSERT INTO Guest (F\_Name, L\_Name, Age, Email\_ID, Phone, Platinum\_Member)

VALUES ('Sai Suma', 'Maguluri', 24, 'saisuma.maguluri@utdallas.edu', '(972)883-2705', 'Yes');

INSERT INTO Guest (F\_Name, L\_Name, Age, Email\_ID, Phone, Platinum\_Member)

VALUES ('John', 'Doe', 69, 'john.doe@gmail.com', '(469)850-5189', 'No');

INSERT INTO Guest (F\_Name, L\_Name, Age, Email\_ID, Phone, Platinum\_Member)

VALUES ('Jacob', 'Foster', 31, 'jacob.foster@gmail.com', '(214)573-7890', 'Yes');

INSERT INTO Guest (F\_Name, L\_Name, Age, Email\_ID, Phone, Platinum\_Member)

VALUES ('John', 'Smith', 26,'john.smith@yahoo.com', '(469)650-9879', 'No');

INSERT INTO Guest (F\_Name, L\_Name, Age, Email\_ID, Phone, Platinum\_Member)

VALUES ('Andrea', 'Jones', 55, 'andreajones@hotmail.com', '(630)639-2914', 'No');

INSERT INTO Guest (F\_Name, L\_Name, Age, Email\_ID, Phone, Platinum\_Member)

VALUES ('Taylor', 'Anderson', 74,'taylor.anderson@gmail.com', '(469)879-0770', 'No');

INSERT INTO Guest (F\_Name, L\_Name, Age, Email\_ID, Phone, Platinum\_Member)

VALUES ('Aman', 'Sharma', 23,'sharmaaman@yahoo.com', '(847)245-0990', 'Yes');

INSERT INTO Guest (F\_Name, L\_Name, Age, Email\_ID, Phone, Platinum\_Member)

VALUES ('Kyle', 'Cunningham', 45,'cunninghamk@gmail.com', '(202)230-8970', 'Yes');

INSERT INTO Guest (F\_Name, L\_Name, Age, Email\_ID, Phone, Platinum\_Member)

VALUES ('Mehnaz', 'Mahmood', 51, 'mmahmood@yahoo.com', '(449)517-9809', 'No');

-- Reservation table

INSERT INTO Reservation (Guest\_ID, Checkin\_Dt, Checkout\_Dt, Res\_Status, No\_of\_Guests, No\_of\_TP\_Tickets, Res\_Date)

VALUES (101, '2022-10-24', '2022-10-27', 'Confirmed', 1, 0, '2022-10-14');

INSERT INTO Reservation (Guest\_ID, Checkin\_Dt, Checkout\_Dt, Res\_Status, No\_of\_Guests, No\_of\_TP\_Tickets, Res\_Date)

VALUES (102, '2022-09-29', '2022-10-03', 'Confirmed', 5, 5, '2022-09-10');

INSERT INTO Reservation (Guest\_ID, Checkin\_Dt, Checkout\_Dt, Res\_Status, No\_of\_Guests, No\_of\_TP\_Tickets, Res\_Date)

VALUES (103, '2022-10-21', '2022-10-26', 'Confirmed', 4, 4, '2022-10-11');

INSERT INTO Reservation (Guest\_ID, Checkin\_Dt, Checkout\_Dt, Res\_Status, No\_of\_Guests, No\_of\_TP\_Tickets, Res\_Date)

VALUES (104, '2022-11-14', '2022-11-19', 'Confirmed', 2, 2, '2022-10-30');

INSERT INTO Reservation (Guest\_ID, Checkin\_Dt, Checkout\_Dt, Res\_Status, No\_of\_Guests, No\_of\_TP\_Tickets, Res\_Date)

VALUES (105, '2022-11-27', '2022-12-01', 'Confirmed', 2, 2, '2022-11-20');

INSERT INTO Reservation (Guest\_ID, Checkin\_Dt, Checkout\_Dt, Res\_Status, No\_of\_Guests, No\_of\_TP\_Tickets, Res\_Date)

VALUES (106, '2022-12-08', '2022-12-12', 'Confirmed', 2, 2, '2022-11-15');

INSERT INTO Reservation (Guest\_ID, Checkin\_Dt, Checkout\_Dt, Res\_Status, No\_of\_Guests, No\_of\_TP\_Tickets, Res\_Date)

VALUES (107, '2022-12-12', '2022-12-15', 'Confirmed', 3, 2, '2022-11-10');

INSERT INTO Reservation (Guest\_ID, Checkin\_Dt, Checkout\_Dt, Res\_Status, No\_of\_Guests, No\_of\_TP\_Tickets, Res\_Date)

VALUES (108, '2022-12-15', '2022-12-18', 'Waiting\_for\_Room Availability', 4, 3, '2022-11-14');

INSERT INTO Reservation (Guest\_ID, Checkin\_Dt, Checkout\_Dt, Res\_Status, No\_of\_Guests, No\_of\_TP\_Tickets, Res\_Date)

VALUES (109, '2022-12-23', '2022-12-25', 'Waiting\_for\_Room Availability', 3, 3, '2022-11-22');

INSERT INTO Reservation (Guest\_ID, Checkin\_Dt, Checkout\_Dt, Res\_Status, No\_of\_Guests, No\_of\_TP\_Tickets, Res\_Date)

VALUES (110, '2022-12-25', '2022-12-30', 'Waiting\_for\_Room Availability', 2, 2, '2022-12-15');

-- Rooms Table

INSERT INTO Rooms (Room\_No, Res\_No, Room\_Category, Room\_Type, Room\_View, Breakfast, Room\_Rate)

VALUES ('401', '10025', 'Standard King', 'Non-Smoking', 'Street View', 'No', '$160');

INSERT INTO Rooms(Room\_No, Res\_No, Room\_Category, Room\_Type, Room\_View, Breakfast, Room\_Rate)

VALUES('302', '10026', 'Standard Queen', 'Smoking', 'Themepark View', 'Yes', '$170');

INSERT INTO Rooms(Room\_No, Res\_No, Room\_Category, Room\_Type, Room\_View, Breakfast, Room\_Rate)

VALUES('203', '10027', 'Standard Twin', 'Non-Smoking', 'Pool View', 'Yes', '$140');

INSERT INTO Rooms(Room\_No, Res\_No, Room\_Category, Room\_Type, Room\_View, Breakfast, Room\_Rate)

VALUES('504', '10028', 'Executive Suite', 'Smoking', 'Pool View', 'Yes', '$225');

INSERT INTO Rooms(Room\_No, Res\_No, Room\_Category, Room\_Type, Room\_View, Breakfast, Room\_Rate)

VALUES('305', '10029', 'Standard Queen', 'Smoking', 'Pool View', 'Yes', '$165');

INSERT INTO Rooms(Room\_No, Res\_No, Room\_Category, Room\_Type, Room\_View, Breakfast, Room\_Rate)

VALUES('206', '10030', 'Standard Twin', 'Non-Smoking', 'Themepark View', 'No', '$135');

INSERT INTO Rooms(Room\_No, Res\_No, Room\_Category, Room\_Type, Room\_View, Breakfast, Room\_Rate)

VALUES('507', '10031', 'Executive Suite', 'Smoking', 'Themepark View', 'Yes', '$230');

INSERT INTO Rooms(Room\_No, Res\_No, Room\_Category, Room\_Type, Room\_View, Breakfast, Room\_Rate)

VALUES('408', '10032', 'Standard King', 'Smoking', 'Garden View', 'Yes', '$180');

INSERT INTO Rooms(Room\_No, Res\_No, Room\_Category, Room\_Type, Room\_View, Breakfast, Room\_Rate)

VALUES('209', '10033', 'Standard Twin', 'Non-Smoking', 'Pool View', 'Yes', '$140');

INSERT INTO Rooms(Room\_No, Res\_No, Room\_Category, Room\_Type, Room\_View, Breakfast, Room\_Rate)

VALUES('510', '10034', 'Executive Suite', 'Smoking', 'Themepark View', 'Yes', '$230');

-- Pricing Table

INSERT INTO Pricing (Discount\_Code, Discount\_Type, Discount\_percent, Theme\_Park\_Price, Room\_Tax\_Rate, TP\_Ticket\_Tax\_Rate)

VALUES ('EB', 'Early Bird', 10, 55, 13, 8.25);

INSERT INTO Pricing (Discount\_Code, Discount\_Type, Discount\_percent, Theme\_Park\_Price, Room\_Tax\_Rate, TP\_Ticket\_Tax\_Rate)

VALUES ('RT', 'Room Theme Park Combo', 20, 55, 13, 8.25);

INSERT INTO Pricing (Discount\_Code, Discount\_Type, Discount\_percent, Theme\_Park\_Price, Room\_Tax\_Rate, TP\_Ticket\_Tax\_Rate)

VALUES ('SC', 'Senior Citizen Discount', 10, 55, 13, 8.25);

INSERT INTO Pricing (Discount\_Code, Discount\_Type, Discount\_percent, Theme\_Park\_Price, Room\_Tax\_Rate, TP\_Ticket\_Tax\_Rate)

VALUES ('PM', 'Platinum Member', 15, 55, 13, 8.25);

INSERT INTO Pricing (Discount\_Code, Discount\_Type, Discount\_percent, Theme\_Park\_Price, Room\_Tax\_Rate, TP\_Ticket\_Tax\_Rate)

VALUES ('T3', 'More than 2 Theme Park Tickets', 5, 55, 13, 8.25);

INSERT INTO Pricing (Discount\_Code, Discount\_Type, Discount\_percent, Theme\_Park\_Price, Room\_Tax\_Rate, TP\_Ticket\_Tax\_Rate)

VALUES ('EXEM', 'Tax Exempt', 0, 55, 0, 0);

INSERT INTO Pricing (Discount\_Code, Discount\_Type, Discount\_percent, Theme\_Park\_Price, Room\_Tax\_Rate, TP\_Ticket\_Tax\_Rate)

VALUES ('VET', 'Veteran', 7, 55, 13, 8.25);

INSERT INTO Pricing (Discount\_Code, Discount\_Type, Discount\_percent, Theme\_Park\_Price, Room\_Tax\_Rate, TP\_Ticket\_Tax\_Rate)

VALUES ('RES', 'Local Texas Resident', 2, 55, 13, 8.25);

INSERT INTO Pricing (Discount\_Code, Discount\_Type, Discount\_percent, Theme\_Park\_Price, Room\_Tax\_Rate, TP\_Ticket\_Tax\_Rate)

VALUES ('RSF', 'Room Service Food Order', 2, 55, 13, 8.25);

INSERT INTO Pricing (Discount\_Code, Discount\_Type, Discount\_percent, Theme\_Park\_Price, Room\_Tax\_Rate, TP\_Ticket\_Tax\_Rate)

VALUES ('CS', 'College Student', 3, 55, 13, 8.25);

-- Payment table

INSERT INTO Payment (Res\_No, Pricing\_ID, Pmt\_Method, Card\_Number, Type\_of\_Card, Name\_on\_Card, Exp\_Date, Pmt\_Status)

VALUES (10025,6,'Credit Card','4044120045673450','Visa','Gasan Elkhodari','04/2026','Paid');

INSERT INTO Payment (Res\_No, Pricing\_ID, Pmt\_Method, Card\_Number, Type\_of\_Card, Name\_on\_Card, Exp\_Date, Pmt\_Status)

VALUES (10026,2,'Debit Card','5002341245783458','Mastercard','Sai Suma Maguluri','05/2025','Paid');

INSERT INTO Payment (Res\_No, Pricing\_ID, Pmt\_Method, Card\_Number, Type\_of\_Card, Name\_on\_Card, Exp\_Date, Pmt\_Status)

VALUES (10027,2,'Credit Card','4232456789762345','Visa','John Doe','04/2023','Paid');

INSERT INTO Payment (Res\_No, Pricing\_ID, Pmt\_Method, Card\_Number, Type\_of\_Card, Name\_on\_Card, Exp\_Date, Pmt\_Status)

VALUES (10028,2,'Credit Card','4908789067534573','Visa','Jacob Foster','09/2026','Balance Due');

INSERT INTO Payment (Res\_No, Pricing\_ID, Pmt\_Method, Card\_Number, Type\_of\_Card, Name\_on\_Card, Exp\_Date, Pmt\_Status)

VALUES (10029,2,'Credit Card','5590898765456723','Mastercard','John Smith','07/2025','Balance Due');

INSERT INTO Payment (Res\_No, Pricing\_ID, Pmt\_Method, Card\_Number, Type\_of\_Card, Name\_on\_Card, Exp\_Date, Pmt\_Status)

VALUES (10030,2,'Credit Card','4890654323455789','Visa','Andrea Jones','12/2026','Balance Due');

INSERT INTO Payment (Res\_No, Pricing\_ID, Pmt\_Method, Card\_Number, Type\_of\_Card, Name\_on\_Card, Exp\_Date, Pmt\_Status)

VALUES (10031,2,'Credit Card','4509897654231257','Visa','Taylor Anderson','11/2025','Balance Due');

INSERT INTO Payment (Res\_No, Pricing\_ID, Pmt\_Method, Card\_Number, Type\_of\_Card, Name\_on\_Card, Exp\_Date, Pmt\_Status)

VALUES (10032,2,'Credit Card','4123578974562345','Visa','Aman Sharma','03/2024','Balance Due');

INSERT INTO Payment (Res\_No, Pricing\_ID, Pmt\_Method, Card\_Number, Type\_of\_Card, Name\_on\_Card, Exp\_Date, Pmt\_Status)

VALUES (10033,2,'Debit Card','4678657089234452','Visa','Kyle Cunningham','05/2024','Balance Due');

INSERT INTO Payment (Res\_No, Pricing\_ID, Pmt\_Method, Card\_Number, Type\_of\_Card, Name\_on\_Card, Exp\_Date, Pmt\_Status)

VALUES (10034,2,'Credit Card','5890786543218764','Mastercard','Mehnaz Mahmood','12/2023','Balance Due');

# Query Source Code

-- 20 SQL Queries

-- Q1. Select all columns and all rows from one table

SELECT \* FROM Guest;

-- Q2. Select five columns and all rows from one table

SELECT Room\_Category, Room\_View, Room\_Type, Breakfast, Room\_Rate FROM Rooms;

-- Q3. Select all columns from all rows from one view

SELECT \* FROM ReservationInfo;

-- Q4. Using a join on 2 tables, select all columns and all rows

SELECT \* FROM Reservation

LEFT OUTER JOIN Guest ON Reservation.Guest\_ID = Guest.Guest\_ID;

-- Q5. Select and order data retrieved from one table

SELECT \* FROM Rooms

ORDER BY Room\_No;

-- Q6. Using a join on 3 tables, select 5 columns from the 3 tables. Use syntax that would limit the output to 10 rows

SELECT Guest.F\_Name, Guest.L\_Name, Reservation.Res\_Status, Rooms.Room\_Type, Rooms.Room\_Rate

FROM Reservation

INNER JOIN Guest ON Reservation.Guest\_ID = Guest.Guest\_ID

INNER JOIN Rooms ON Rooms.Res\_No = Reservation.Res\_No

LIMIT 10;

-- Q7. Select distinct rows using joins on 3 tables

SELECT DISTINCT Rooms.Room\_Category, Rooms.Room\_View, Rooms.Room\_Type, Rooms.Room\_Rate, Payment.Pricing\_ID

FROM Rooms

INNER JOIN Reservation ON Rooms.Res\_no = Reservation.Res\_no

INNER JOIN Payment ON Reservation.Res\_no = Payment.Res\_no;

-- Q8. Use GROUP BY and HAVING in a select statement using one or more tables

SELECT Room\_Category, Room\_View, Room\_Type, Breakfast, Room\_Rate

FROM Rooms

GROUP BY Room\_Rate, Room\_Category, Room\_View, Room\_Type, Breakfast

HAVING Room\_View = 'Themepark View'

ORDER BY Room\_Rate;

-- Q9.Use IN clause to select data from one or more tables

SELECT \* FROM Pricing

WHERE Discount\_percent IN (10,15,20);

-- Q10.Select length of one column from one table

SELECT L\_Name, Length(L\_Name) from Guest;

-- Q11.Delete one record from one table. Use select statements to demonstrate the table contents before and after the DELETE statement. Make sure you use ROLLBACK afterwards so that the data will not be physically removed

COMMIT;

SELECT \* FROM Payment;

DELETE FROM Payment WHERE Pricing\_ID = 6;

SELECT \* FROM Payment;

ROLLBACK;

-- Q12.Update one record from one table. Use select statements to demonstrate the table contents before and after the UPDATE statement. Make sure you use ROLLBACK afterwards so that the data will not be physically removed

START TRANSACTION;

-- Your UPDATE statement

UPDATE Pricing

SET Discount\_Type = 'Military Service Discount'

WHERE Pricing\_ID = '7';

-- Optionally, you can SELECT to verify changes

SELECT \* FROM Pricing WHERE Pricing\_ID = '7';

SELECT \* FROM Pricing;

-- If everything is correct, you can COMMIT the changes

COMMIT;

-- If there's an issue and you want to discard changes, you can ROLLBACK

ROLLBACK;

-- Q13. List all guests whose Reservation is in Confirmed Status. List their IDs, Reservation Number, First Name, Last Name, Room Number, Payment Status and Reservation Status

SELECT Guest.Guest\_ID, Reservation.Res\_No, Guest.F\_Name, Guest.L\_Name, Rooms.Room\_No, Payment.Pmt\_Status, Reservation.Res\_Status

FROM Reservation

INNER JOIN Rooms ON Rooms.Res\_No = Reservation.Res\_No

INNER JOIN Guest ON Guest.Guest\_ID = Reservation.Guest\_ID

INNER JOIN Payment ON Reservation.Res\_No = Payment.Res\_No

WHERE Reservation.Res\_Status = 'Confirmed';

-- Q14. Write a query to assist the front desk in order to prepare a list of all guests whose room rate is at least $180. The front desk would like to leave a small complimentary gift with a customized note in their room. Display their Room Number, First Name, Last Name, Room Rate, Payment Status

SELECT Rooms.Room\_No, Guest.F\_Name, Guest.L\_Name, Rooms.Room\_Rate, Payment.Pmt\_Status from Rooms

INNER JOIN Reservation ON Rooms.Res\_No = Reservation.Res\_No

INNER JOIN Payment ON Reservation.Res\_No = payment.Res\_No

INNER JOIN Guest on Guest.Guest\_ID = Reservation.Guest\_ID

WHERE Rooms.Room\_Rate >= '$180'

ORDER BY Rooms.Room\_No;

-- Q15. Hotel DBF 12's Marketing Team reached out to get a list of most popular room categories that get booked the most. Write a query to assist the marketing team. List the Room Category and the Count of the Ones that get booked the most.

SELECT q1.Room\_Category, q1.num\_occurrences

FROM (

SELECT Rooms.Room\_Category, COUNT(\*) AS num\_occurrences

FROM Rooms

GROUP BY Rooms.Room\_Category

) q1, (

SELECT MAX(q2.num\_occurrences) AS max\_occurrences

FROM (

SELECT Rooms.Room\_Category, COUNT(\*) AS num\_occurrences

FROM Rooms

GROUP BY Rooms.Room\_Category

) q2

) q3

WHERE q1.num\_occurrences = q3.max\_occurrences;

-- Q16. List all reservations where the Room Theme Park Combo Discount has been applied and a Mastercard will be used for Payment

SELECT Reservation.Res\_No, Guest.F\_Name, Guest.L\_Name, Reservation.Res\_Status, Payment.Type\_of\_Card, Pricing.Discount\_Code

FROM Guest

INNER JOIN Reservation ON Guest.Guest\_ID = Reservation.Guest\_ID

INNER JOIN Payment ON Reservation.Res\_no = Payment.Res\_No

INNER JOIN Pricing on Payment.Pricing\_ID = Pricing.Pricing\_ID

WHERE Payment.Type\_Of\_Card = 'Mastercard'

AND Pricing.Discount\_Code = 'RT'

ORDER BY Res\_No;

-- Q17. List all the platinum members who have made reservation for atleast 3 people and have purchased at least 2 Theme Park tickets

Select Guest.Guest\_ID, Guest.F\_Name, Guest.L\_Name, Reservation.No\_of\_Guests, Reservation.No\_of\_TP\_Tickets, Guest.Platinum\_Member

FROM Guest

INNER JOIN Reservation ON Guest.Guest\_ID = Reservation.Guest\_ID

WHERE Guest.Platinum\_Member = 'Yes'

AND Reservation.No\_of\_Guests >= 3

AND Reservation.No\_of\_TP\_Tickets >= 2

ORDER BY Guest\_ID;

-- Q18. Write a query to assist the front desk staff of DBF 12 to calculate room total, theme park total and final amt before and after discount for all the hotel guests.

CREATE TABLE payment\_temp AS

SELECT

a.\*,

(b.No\_of\_Nights) \* CAST(SUBSTRING(c.Room\_Rate, 2, 3) AS DECIMAL) AS room\_total,

(b.No\_of\_TP\_Tickets \* d.Theme\_Park\_Price) AS Theme\_park\_total

FROM

payment a

LEFT OUTER JOIN reservation b ON a.RES\_NO = b.RES\_NO

LEFT OUTER JOIN rooms c ON b.RES\_NO = c.RES\_NO

LEFT OUTER JOIN pricing d ON a.pricing\_id = d.pricing\_id;

SELECT

a.PMT\_ID,

a.RES\_NO,

a.PRICING\_ID,

a.ROOM\_TOTAL,

a.THEME\_PARK\_TOTAL,

Room\_Tax\_Rate,

TP\_Ticket\_Tax\_Rate,

(a.room\_total \* (1 + (b.Room\_Tax\_Rate / 100))) + (a.Theme\_park\_total \* (1 + (b.TP\_Ticket\_Tax\_Rate / 100))) AS final\_amt\_bef\_disc,

((a.room\_total \* (1 + (b.Room\_Tax\_Rate / 100))) + (a.Theme\_park\_total \* (1 + (b.TP\_Ticket\_Tax\_Rate / 100)))) \* (1 - (b.Discount\_percent / 100)) AS final\_amt\_after\_disc

FROM

payment\_temp a

LEFT OUTER JOIN pricing b ON a.pricing\_id = b.pricing\_id;

-- Q19. One of the hotel guest Mr. Gasan Elkhodari showed up to the hotel front desk with a tax exempt certificate mentioning that the purpose of his stay at the hotel is for attending a conference nearby and as a state employee of University of Texas at Dallas, he should be exempted from taxes. Please write a query to assist the front desk staff for calculating Room total for Mr. Elkhodari and more guests like him in the future who would present a valid tax exempt certificate to the hotel front desk.

SELECT A.res\_no, A.f\_name, A.l\_name, B.discount\_code, A.room\_rate, B.room\_total

FROM (

SELECT \*

FROM (

SELECT guest.guest\_id AS g1\_id, guest.f\_name, guest.l\_name

FROM guest

JOIN reservation ON guest.guest\_id = reservation.guest\_id

) AS A1,

(

SELECT reservation.guest\_id AS g2\_id, reservation.res\_no, rooms.room\_rate

FROM reservation

JOIN rooms ON reservation.res\_no = rooms.res\_no

) AS A2

WHERE A1.g1\_id = A2.g2\_id

) AS A,

(

SELECT T.res\_no, pt.room\_total, T.discount\_code

FROM payment\_temp AS pt,

(

SELECT pricing\_id, res\_no, discount\_code

FROM (

SELECT res\_no, discount\_code, pricing\_id

FROM pricing

JOIN (

SELECT reservation.res\_no, pricing\_id AS res\_pid

FROM reservation, payment

WHERE reservation.res\_no = payment.res\_no

) AS P1 ON res\_pid = pricing\_id

) AS P2

WHERE discount\_code = 'EXEM'

) AS T

WHERE pt.res\_no = T.res\_no

) AS B

WHERE A.res\_no = B.res\_no;

-- Q20. For people who availed Room theme park combo, find out how many platinum members are spending more than the average amount

create table payment\_temp1 as

select a.\*,Room\_Tax\_Rate,TP\_Ticket\_Tax\_Rate,(a.room\_total \* (1+(b.Room\_Tax\_Rate/100)))+(a.Theme\_park\_total\*(1+(b.TP\_Ticket\_Tax\_Rate/100))) as final\_amt\_bef\_disc,

((a.room\_total \* (1+(b.Room\_Tax\_Rate/100)))+(a.Theme\_park\_total\*(1+(b.TP\_Ticket\_Tax\_Rate/100))))\*(1 - (b.Discount\_percent/100)) as final\_amt\_after\_disc

from payment\_temp a left outer join pricing b

on a.pricing\_id = b.pricing\_id;

select a.platinum\_member,count(a.guest\_id) as COUNT\_OF\_CUSTOMERS\_WHO\_SPEND\_ABOVE\_AVERAGE

from guest a left outer join reservation b

on a.guest\_id = b.guest\_id

left outer join payment\_temp1 c

on b.res\_no = c.res\_no

left outer join pricing d

on c.pricing\_id = d.pricing\_id

where d.discount\_type = 'Room Theme Park Combo' AND

c.FINAL\_AMT\_AFTER\_DISC > (select avg(FINAL\_AMT\_AFTER\_DISC) from payment\_temp1 where pricing\_id in (select pricing\_id from pricing where discount\_type = 'Room Theme Park Combo'))

group by a.platinum\_member;

DROP TABLE payment\_temp;

DROP TABLE payment\_temp1;

# DDL Output

Trigger TR\_GUEST dropped. Trigger TR\_RES dropped.

Trigger TR\_ROOMS dropped. Trigger TR\_PRICING dropped. Trigger TR\_PAYMENT dropped. Sequence SEQ\_GUEST\_ID dropped. Sequence SEQ\_RES\_NO dropped.

Sequence SEQ\_PRICING\_ID dropped. Sequence SEQ\_PMT\_ID dropped.

View GUESTINFO dropped.

View RESERVATIONINFO dropped. View ROOMSINFO dropped.

View PRICINGINFO dropped. View PAYMENTINFO dropped.

Index IDX\_GUEST\_FNAME dropped. Index IDX\_GUEST\_AGE dropped.

Index IDX\_GUEST\_PLATINUM\_MEMBERSHIP dropped. Index IDX\_RESERVATION\_NO\_OF\_NIGHTS dropped.

Index IDX\_RESERVATION\_NO\_OF\_TP\_TICKETS dropped. Index IDX\_ROOM\_VIEW dropped.

Index IDX\_ROOM\_RATE dropped.

Index IDX\_PRICING\_DISCOUNT dropped.

Index IDX\_PRICING\_DISCOUNT\_TYPE dropped. Index IDX\_PAYMENT\_NAME\_ON\_CARD dropped. Index IDX\_PAYMENT\_CARD\_NUMBER dropped.

Index IDX\_PAYMENT\_RES\_NO dropped. Index IDX\_PAYMNET\_PRICING\_ID dropped. Table PAYMENT dropped.

Table PRICING dropped. Table ROOMS dropped.

Table RESERVATION dropped. Table GUEST dropped.

Table GUEST created.

Table RESERVATION created. Table ROOMS created.

Table PRICING created.

Table PAYMENT created.

Index IDX\_GUEST\_FNAME created. Index IDX\_GUEST\_AGE created.

Index IDX\_GUEST\_PLATINUM\_MEMBERSHIP created. Index IDX\_RESERVATION\_NO\_OF\_NIGHTS created.

Index IDX\_RESERVATION\_NO\_OF\_TP\_TICKETS created. Index IDX\_GUEST\_ID created.

Index IDX\_ROOM\_VIEW created. Index IDX\_ROOM\_RATE created.

Index IDX\_PRICING\_DISCOUNT created.

Index IDX\_PRICING\_DISCOUNT\_TYPE created. Index IDX\_PAYMENT\_NAME\_ON\_CARD created. Index IDX\_PAYMENT\_CARD\_NUMBER created. Index IDX\_PAYMENT\_RES\_NO created.

Index IDX\_PAYMNET\_PRICING\_ID created. Table GUEST altered.

Table RESERVATION altered. Table ROOMS altered.

Table PRICING altered.

Table PAYMENT altered.

View GUESTINFO created.

View RESERVATIONINFO created. View ROOMSINFO created.

View PRICINGINFO created. View PAYMENTINFO created.

Sequence SEQ\_GUEST\_ID created. Sequence SEQ\_RES\_NO created.

Sequence SEQ\_PRICING\_ID created. Sequence SEQ\_PMT\_ID created.

Trigger TR\_GUEST compiled Trigger TR\_RES compiled Trigger TR\_ROOMS compiled Trigger TR\_PRICING compiled Trigger TR\_PAYMENT compiled

# DML Output

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

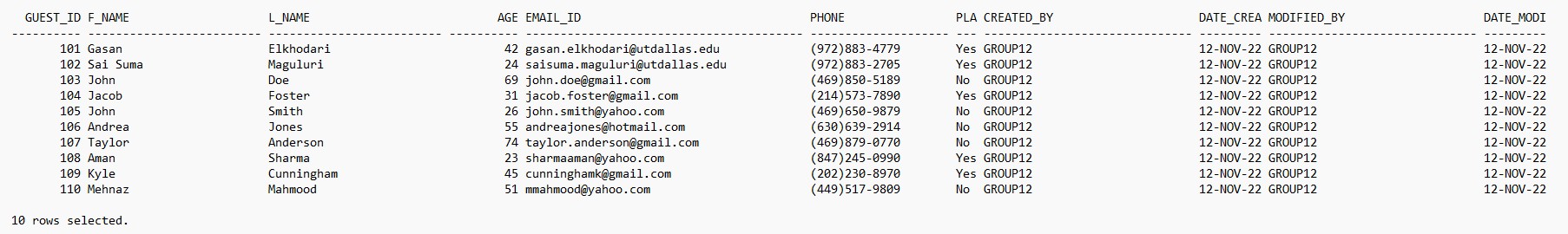
1 row inserted.

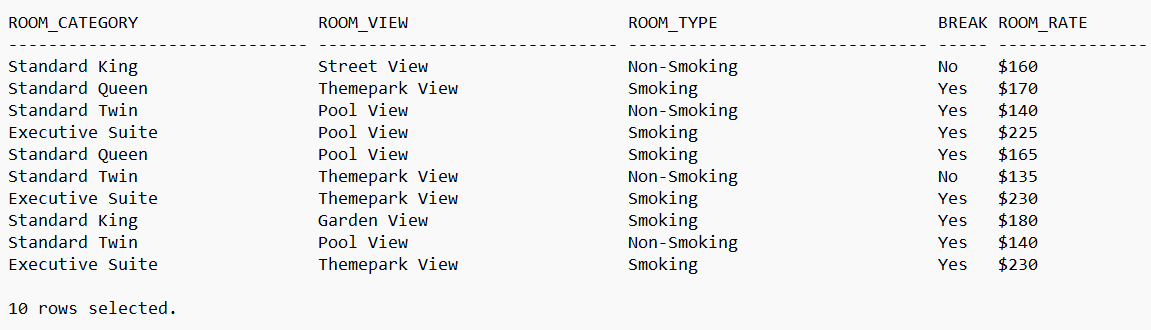
1 row inserted.

1 row inserted.

# Query Output

Q1:



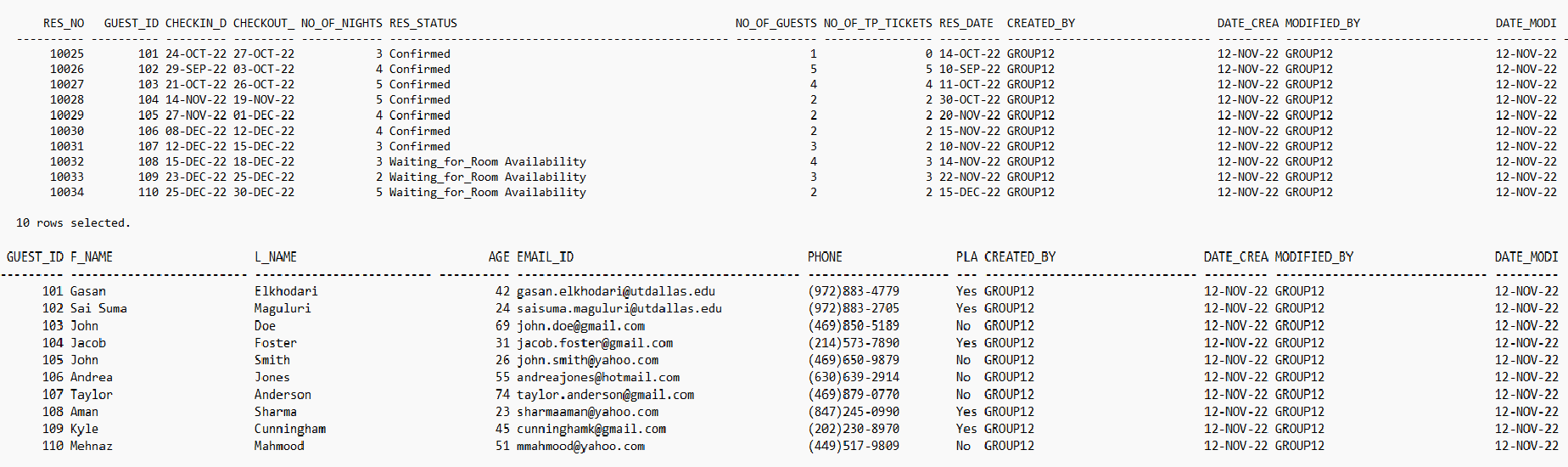
Q2:

Q3:

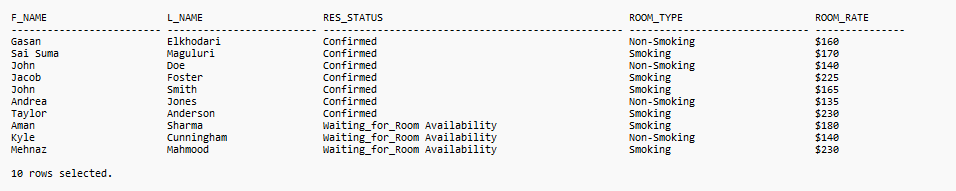
RES\_NO GUEST\_ID NO\_OF\_NIGHTS NO\_OF\_TP\_TICKETS

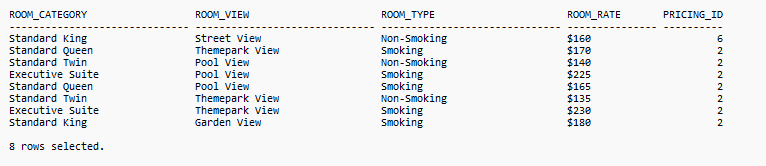
|  |  |  |  |
| --- | --- | --- | --- |
| 10025 | 101 | 3 | 0 |
| 10026 | 102 | 4 | 5 |
| 10027 | 103 | 5 | 4 |
| 10028 | 104 | 5 | 2 |
| 10029 | 105 | 4 | 2 |
| 10030 | 106 | 4 | 2 |
| 10031 | 107 | 3 | 2 |
| 10032 | 108 | 3 | 3 |
| 10033 | 109 | 2 | 3 |
| 10034 | 110 | 5 | 2 |

10 rows selected.

Q4:

Q5.

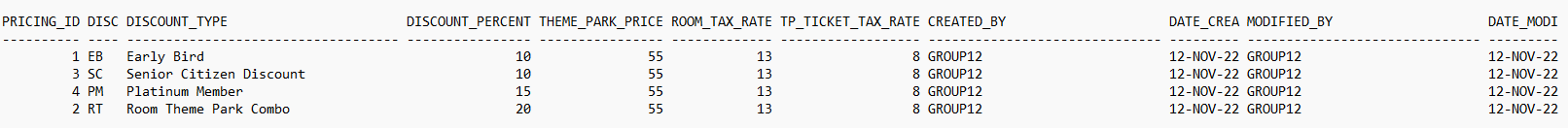
Q6.

Q7.

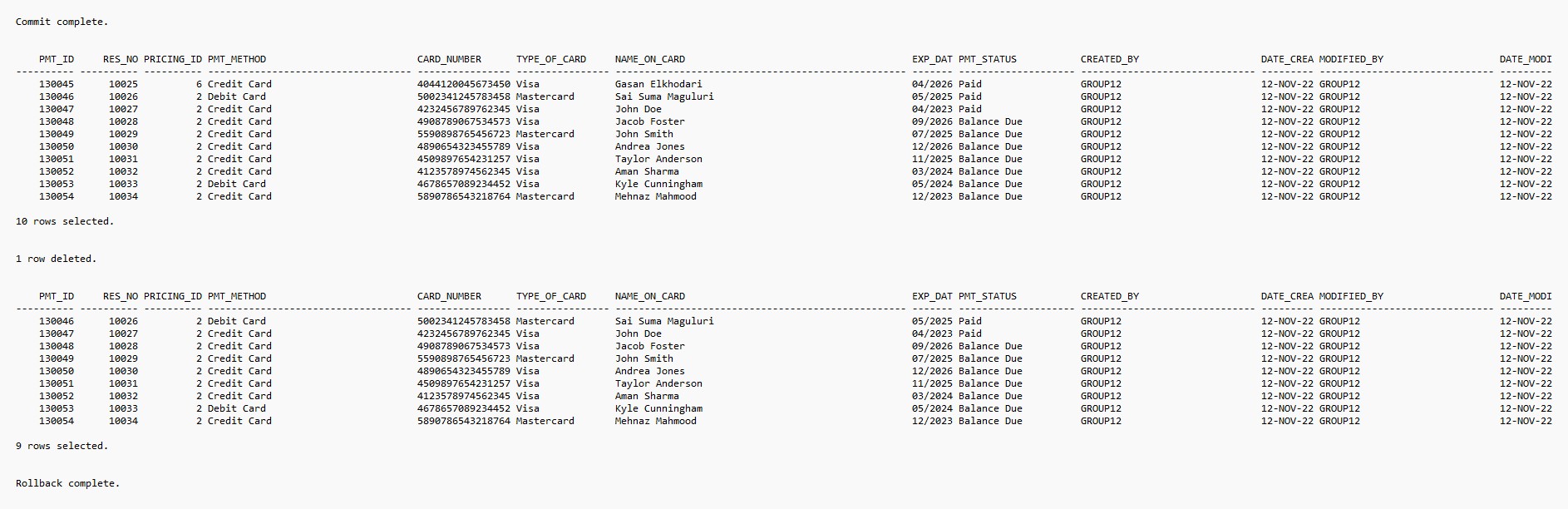
Q8.

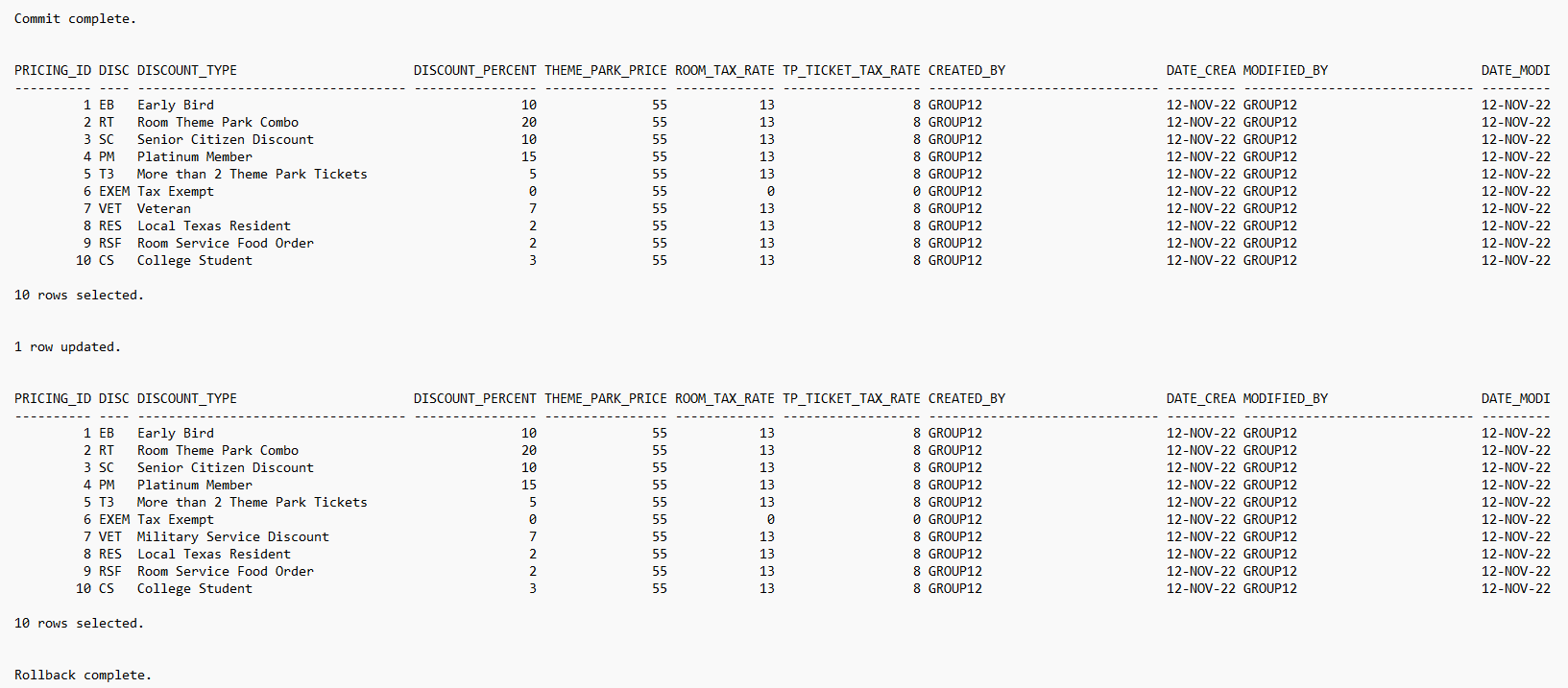
ROOM\_CATEGORY ROOM\_VIEW ROOM\_TYPE BREAK ROOM\_RATE

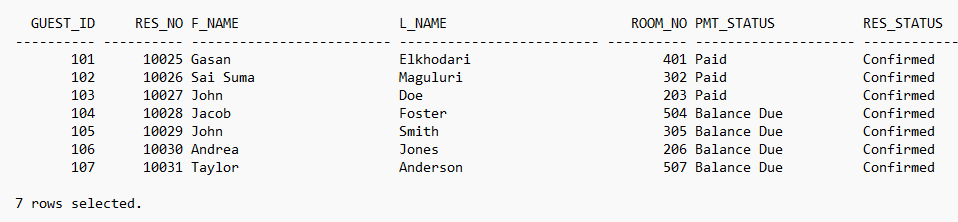
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Standard Twin | Themepark View | Non-Smoking | No | $135 |
| Standard Queen | Themepark View | Smoking | Yes | $170 |
| Executive Suite | Themepark View | Smoking | Yes | $230 |
| Q9. |  |  |  |  |



|  |  |
| --- | --- |
| Q10. L\_NAME | LENGTH(L\_NAME) |
| Elkhodari | 9 |
| Maguluri | 8 |
| Doe | 3 |
| Foster | 6 |
| Smith | 5 |
| Jones | 5 |
| Anderson | 8 |
| Sharma | 6 |
| Cunningham | 10 |
| Mahmood | 7 |
| 10 rows selected. |  |

Q11.

Q12.

Q13.

Q14.

ROOM\_NO F\_NAME L\_NAME ROOM\_RATE PMT\_STATUS

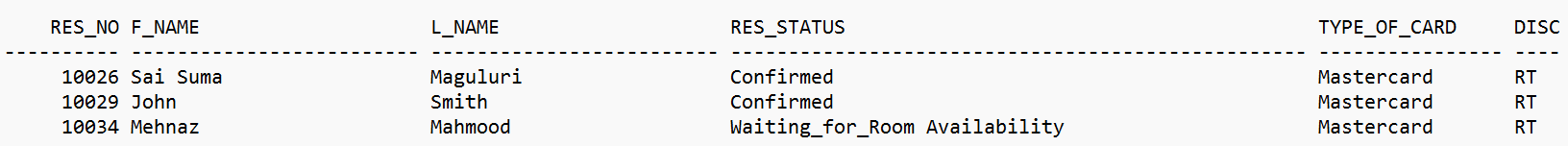
|  |  |  |  |
| --- | --- | --- | --- |
| 408 Aman | Sharma | $180 | Balance Due |
| 504 Jacob | Foster | $225 | Balance Due |
| 507 Taylor | Anderson | $230 | Balance Due |
| 510 Mehnaz | Mahmood | $230 | Balance Due |

Q15.

ROOM\_CATEGORY NUM\_OCCURRENCES

Executive Suite 3

Standard Twin 3

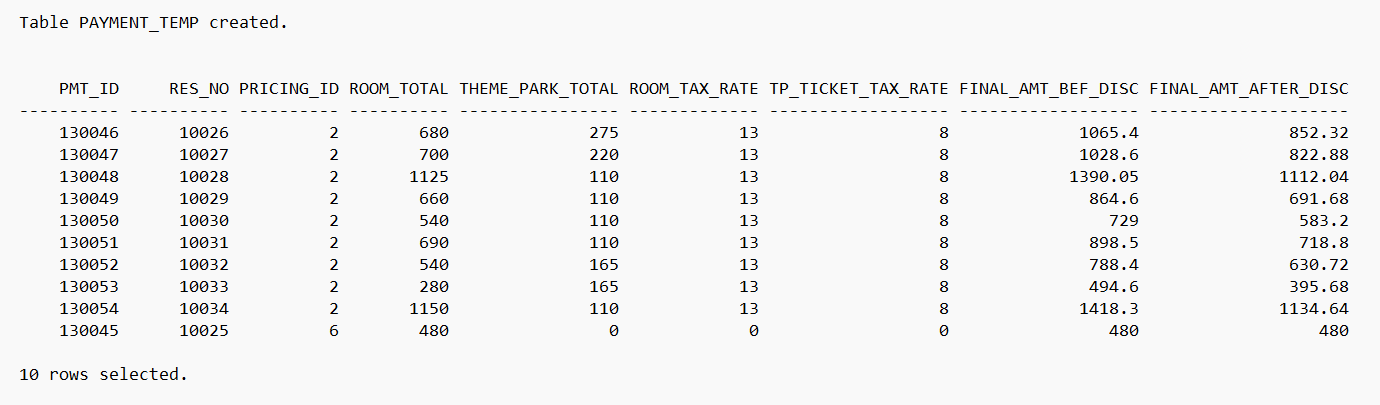
Q16.

Q17.

GUEST\_ID F\_NAME L\_NAME NO\_OF\_GUESTS NO\_OF\_TP\_TICKETS PLA

102 Sai Suma Maguluri 5 5 Yes

1. Aman Sharma 4 3 Yes
2. Kyle Cunningham 3 3 Yes

Q18.

Q19.

RES\_NO F\_NAME L\_NAME DISC ROOM\_RATE ROOM\_TOTAL

10025 Gasan Elkhodari EXEM $160 480

Q20.

Table PAYMENT\_TEMP1 created.

PLA COUNT\_OF\_CUSTOMERS\_WHO\_SPEND\_ABOVE\_AVERAGE

Yes 2

No 2