Hotel Database Management System

Toronto Metropolitan University

Yijuan Huang - 501160292

Nigel Siddeley - 501186392

CPS510: Database Systems I

Professor Abhari Abdolreza

Section 5

November 30, 2024

Table Of Contents

Description

Hotels have always kept ledgers for bookkeeping purposes, as it is self-evident that a hotel should be able to keep track of guests and current room reservations to provide service to incipient and current guests. The near universal proliferation of computer technology over the past few decades has seen most hotels transition from cumbersome physical ledgers for storing information to computerized databases. While a simple excel sheet may suffice for a smaller motel, larger hotels will have to rely on more comprehensive data models to keep track of rooms, availability, employee status, guests and numerous other factors. This complexity is especially pronounced in chain hotels, which could have hundreds of different locations spread over

numerous nations and with employee counts in the thousands, all of which will require detailed payroll information. This requires an accessible and logical database to keep note of all a hotel chain's operations and employees. This database should be able to output both individualized items or groupings based on queries given by the user and should be available for use by any high enough manager in the hotel firm to manage a hotel.

By necessity, any hotel management system will have to keep track of a hotel's various rooms, maintaining detailed information about a room's number, hotel location, suite, occupancy status, price, and other details such as the number of beds and requested client accommodations. All this information should be displayed in a concise and logical manner for the benefit of the user. The primary key to a room is its number, which should be unique for a given hotel. There should also be the ability to view attached notes for a given room to see what services and cleaning must be done, such as restocking a minibar or repainting of the walls. Rooms rendered unavailable for reasons other than a guest should also have a system to explain why the room cannot be booked.

Guests who will be staying at a hotel will need to also be accounted for, with each being given a

unique customer ID as a primary attribute with other accompanying information such as name, age, room occupancy, length of stay and any special accommodations needed. A manager should
also be able to blacklist troublesome guests to prevent them from utilizing the services of a hotel.

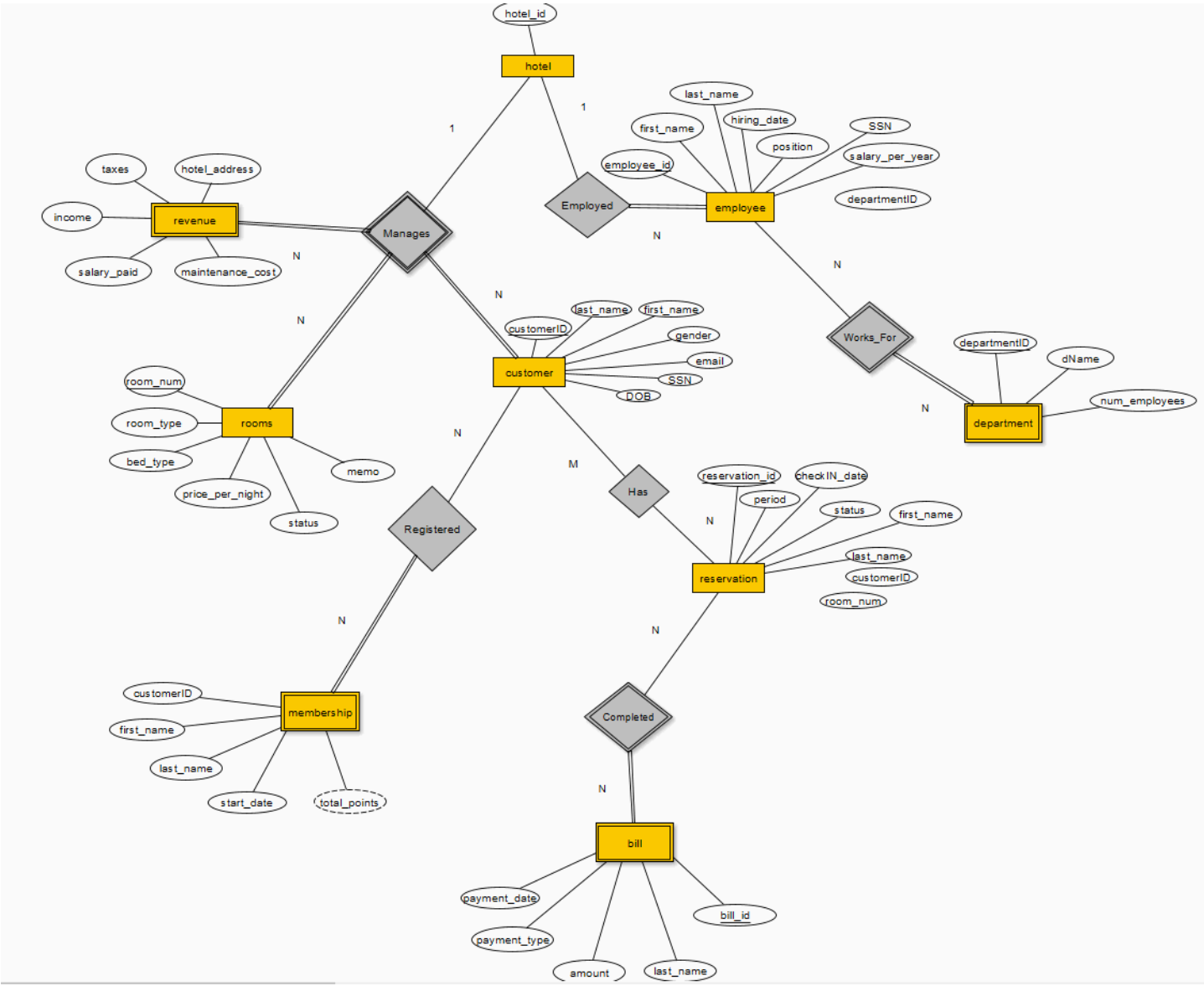
Employees will also be assigned a unique ID for identification purposes, and among other things their info will require their hiring date, name, age, position and salary.

A manager should also be allowed to take a glance at a hotel's revenue and expenses, with relations indicating things such as income, salaries, taxes, operational overhead and maintenance
costs, and other unexpected expenses such as lawsuits and repairs. This information should be displayed in a segregated view for individual revenues and expenses but also sum to an overall cash flow figure to see the financial health of the hotel.

Incoming reservations should also be managed with a unique identifying number as a primary key. A reservation should be displayed along with its length, price, starting date and ending date, as well as the guest(s) who called for the reservation.

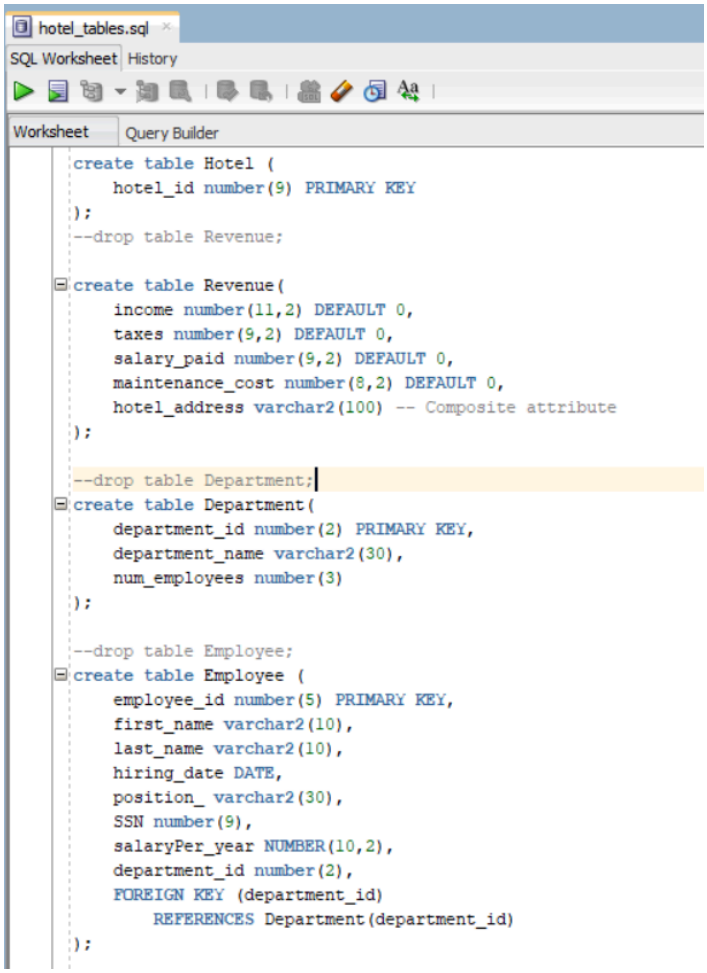
With this framework in mind, we hope to model an effective hotel DBMS that allows a manager to get a complete picture of their hotel's current operations so that they may undertake business decisions in a more sophisticated manner armed with accurate information.

ER Diagram


hotel_id
hotel
1
1
last_name
first_name
hiring_date
SSN
position
salary_per_year
employee_id
departmentID
taxes
hotel_address
income
revenue
Manages
Employed
employee
salary_paid
maintenance_cost
N
N
N
N
N
last_name
first_name
customerID
gender
email
SSN
DOB
customer
Works_For
departmentID
dName
num_employees
department
room_num
room_type
bed_type
price_per_night
status
memo
rooms
N
M
Has
reservation_id
checkIN_date
period
status
first_name
last_name
customerID
room_num
reservation
Registered
N
N
N
customerID
first_name
last_name
start_date
total_points
membership
Completed
N
bill
payment_date
payment_type
amount
last_name
bill_id

Schema Design And Database Construction

Create tables

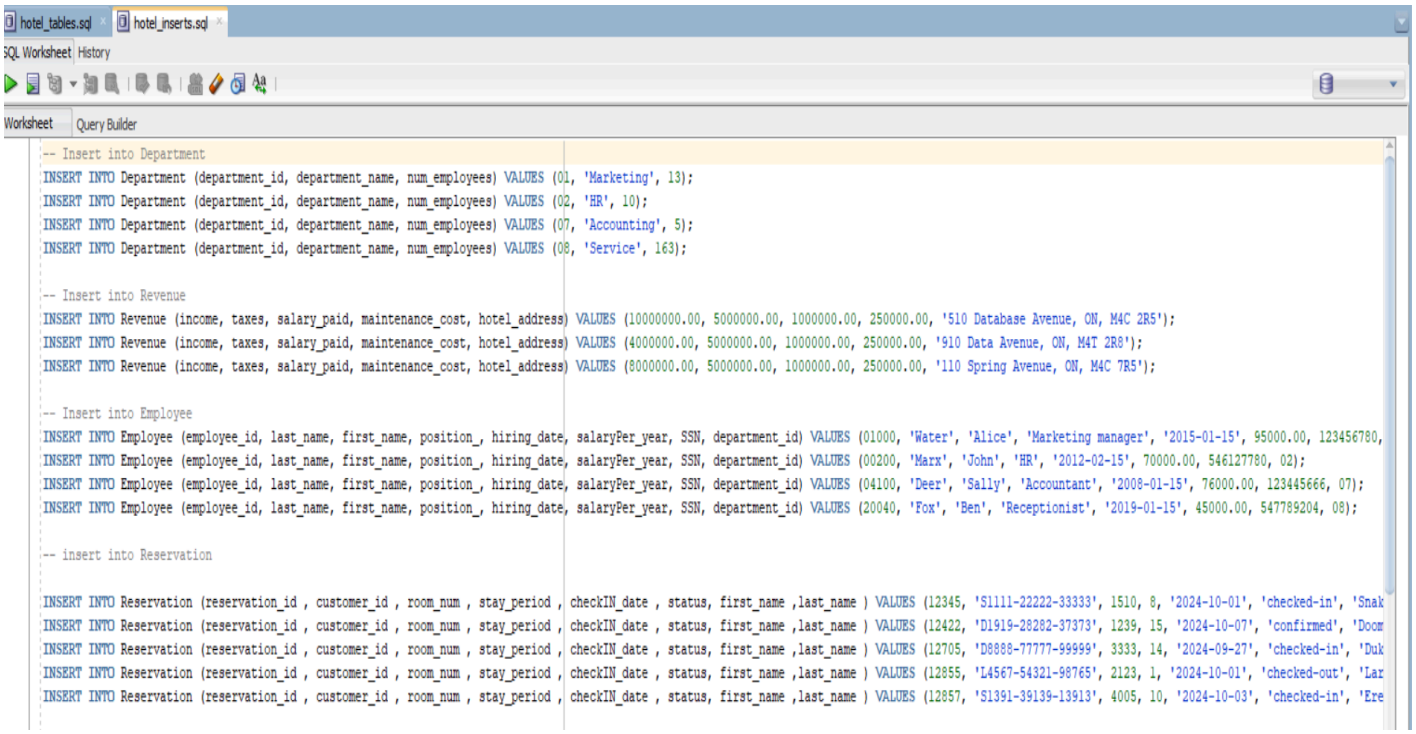



```
create table Hotel (
    hotel_id number(9) PRIMARY KEY
);
--drop table Revenue;

create table Revenue(
    income number(11,2) DEFAULT 0,
    taxes number(9,2) DEFAULT 0,
    salary_paid number(9,2) DEFAULT 0,
    maintenance_cost number(8,2) DEFAULT 0,
    hotel_address varchar2(100) -- Composite attribute
);

--drop table Department;
create table Department(
    department_id number(2) PRIMARY KEY,
    department_name varchar2(30),
    num_employees number(3)
);

--drop table Employee;
create table Employee (
    employee_id number(5) PRIMARY KEY,
    first_name varchar2(10),
    last_name varchar2(10),
    hiring_date DATE,
    position_ varchar2(30),
    SSN number(9),
    salaryPer_year NUMBER(10,2),
    department_id number(2),
    FOREIGN KEY (department_id)
        REFERENCES Department(department_id)
);
```

```
create table Customer(
    customer_id varchar2(20) PRIMARY KEY,
    first_name varchar2(10),
    last_name varchar2(10),
    gender varchar2(10),
    email varchar(40),
    SSN number(9),
    DOB DATE NOT NULL
);

create table Reservation(
    reservation_id varchar(30) PRIMARY KEY,
    customer_id varchar2(20),
    room_num number(6),
    stay_period INT,
    checkIN_date DATE,
    status char(20)  CHECK (status IN ('confirmed','checked-in','checked-out')),
    first_name varchar2(10),
    last_name varchar2(10)
);
```

```sql
create table Bill(
    bill_id number(9) PRIMARY KEY,
    last_name varchar2(10),
    amount number(10,2),
    payment_type char(10),
    payment_date DATE
);

create table Room(
    room_num number(6) PRIMARY KEY,
    bed_type char(30),
    room_type char(30),
    price_per_night INT,
    status varchar2(20) CHECK (status IN ('available','occupied','under maintenance')), --This constraint ensur
    memo varchar2(100)
);

create table Membership(
    customer_id varchar2(20),
    first_name varchar2(10),
    last_name varchar2(10),
    start_date DATE,
    total_points number(7),
    FOREIGN KEY (customer_id)
        REFERENCES Customer(customer_id)
);
```

Populate tables with data



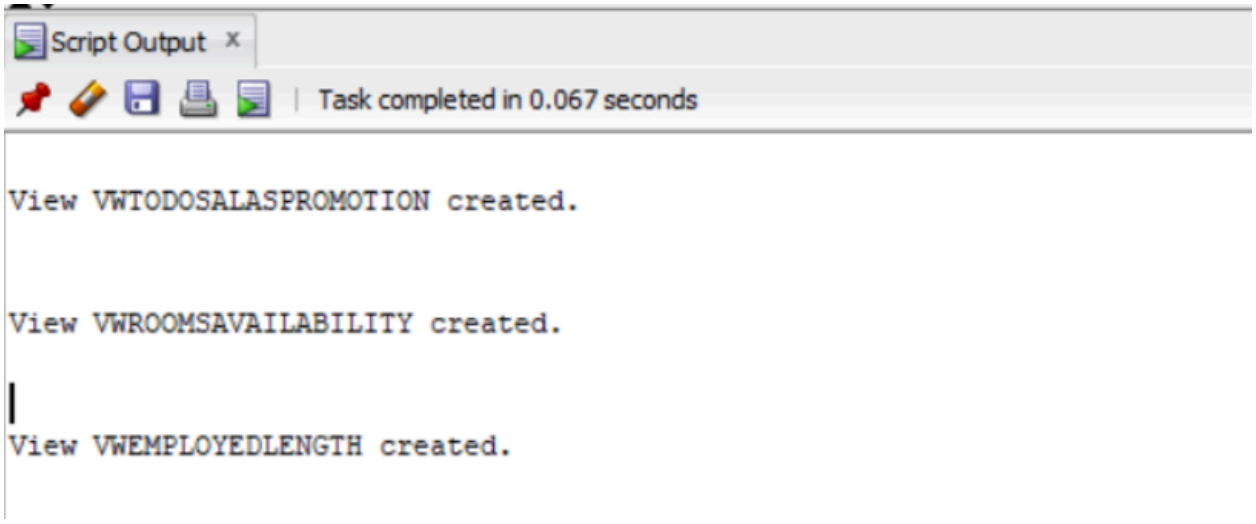
```sql
-- insert into Rooms

INSERT INTO Room (room_num, bed_type, room_type, price_per_night, status) VALUES (1510, 'Double', 'Single Room', 60, 'occupied');
INSERT INTO Room (room_num, bed_type, room_type, price_per_night, status) VALUES (1239, 'Double', 'Double Room', 100, 'occupied');
INSERT INTO Room (room_num, bed_type, room_type, price_per_night, status) VALUES (3333, 'King', 'Deluxe Room', 300, 'occupied');
INSERT INTO Room (room_num, bed_type, room_type, price_per_night, status) VALUES (2123, 'Queen', 'Queen Room', 150, 'available');
INSERT INTO Room (room_num, bed_type, room_type, price_per_night, status) VALUES (4005, 'King', 'Presidential Suite', 2000, 'occupied');
INSERT INTO Room (room_num, bed_type, room_type, price_per_night, status) VALUES (1414, 'Double', 'Single Room', 60, 'under maintenance');
INSERT INTO Room (room_num, bed_type, room_type, price_per_night, status) VALUES (2250, 'King', 'King Room', 200, 'available');
INSERT INTO Room (room_num, bed_type, room_type, price_per_night, status) VALUES (4001, 'Queen', 'Suite', 1000, 'occupied');
INSERT INTO Room (room_num, bed_type, room_type, price_per_night, status) VALUES (3346, 'Double', 'Studio', 400, 'under maintenance');

-- insert into Bill
INSERT INTO Bill (bill_id, last_name, amount, payment_type, payment_date) VALUES (000000011, 'White', 22568.52, 'Cash', '2004-02-15');
INSERT INTO Bill (bill_id, last_name, amount, payment_type, payment_date) VALUES (001234007, 'Bond', 1507.29, 'Credit', '1998-06-01');
INSERT INTO Bill (bill_id, last_name, amount, payment_type, payment_date) VALUES (000013139, 'Jaeger', 663.13, 'Debit', '2013-08-12');
INSERT INTO Bill (bill_id, last_name, amount, payment_type, payment_date) VALUES (001234567, 'Smith', 213.57, 'Credit', '2024-09-29');
INSERT INTO Bill (bill_id, last_name, amount, payment_type, payment_date) VALUES (000007824, 'Andrews', 10000.01, 'Credit', '2023-07-15');
```

```sql
-- insert into Customer
INSERT INTO Customer (customer_id, first_name, last_name, gender, email, ssn, dob) VALUES ('S1111-22222-33333', 'Snake', 'Plisken', 'Male', 'metalgearman@gmail.com', 132435467, '1990-05-05');
INSERT INTO Customer (customer_id, first_name, last_name, gender, email, ssn, dob) VALUES ('D8888-77777-99999', 'Duke', 'Nukem', 'Male', 'thenukeduke@gmail.com', 555555555, '1989-07-29');
INSERT INTO Customer (customer_id, first_name, last_name, gender, email, ssn, dob) VALUES ('S4125-59190-59326', 'John', 'Smith', 'Male', 'thejohnsmith12@gmail.com', 332130382, '1955-02-25');
INSERT INTO Customer (customer_id, first_name, last_name, gender, email, ssn, dob) VALUES ('S1391-39139-13913', 'Eren', 'Jaeger', 'Male', 'titanman@paradis.com', 132000139, '1990-03-30');
INSERT INTO Customer (customer_id, first_name, last_name, gender, email, ssn, dob) VALUES ('K1234-12345-67890', 'Sally', 'Wrinkle', 'Female', 'imoldl23@fossil.com', 000000001, '1812-01-01');
INSERT INTO Customer (customer_id, first_name, last_name, gender, email, ssn, dob) VALUES ('11111-11111-11111', 'Nigel', 'Siddeley', 'Male', 'nsiddeley@gmail.com', 123123123, '2004-02-15');

-- insert into Membership
INSERT INTO Membership (customer_id, first_name, last_name, start_date, total_points) VALUES ('S1111-22222-33333', 'Snake', 'Plisken', '2022-01-25', 5055);
INSERT INTO Membership (customer_id, first_name, last_name, start_date, total_points) VALUES ('D8888-77777-99999', 'Duke', 'Nukem', '2023-11-11', 8888);
INSERT INTO Membership (customer_id, first_name, last_name, start_date, total_points) VALUES ('S4125-59190-59326', 'John', 'Smith', '2000-05-05', 12345);
INSERT INTO Membership (customer_id, first_name, last_name, start_date, total_points) VALUES ('S1391-39139-13913', 'Eren', 'Jaeger', '2017-07-10', 1391);
INSERT INTO Membership (customer_id, first_name, last_name, start_date, total_points) VALUES ('K1234-12345-67890', 'Sally', 'Wrinkle', '1900-02-02', 999999);
```

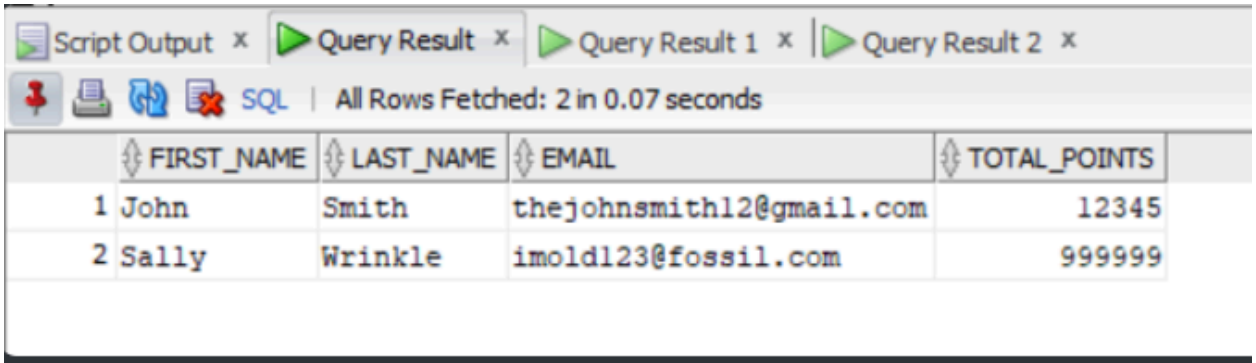
Views And Simple Queries

Creating views

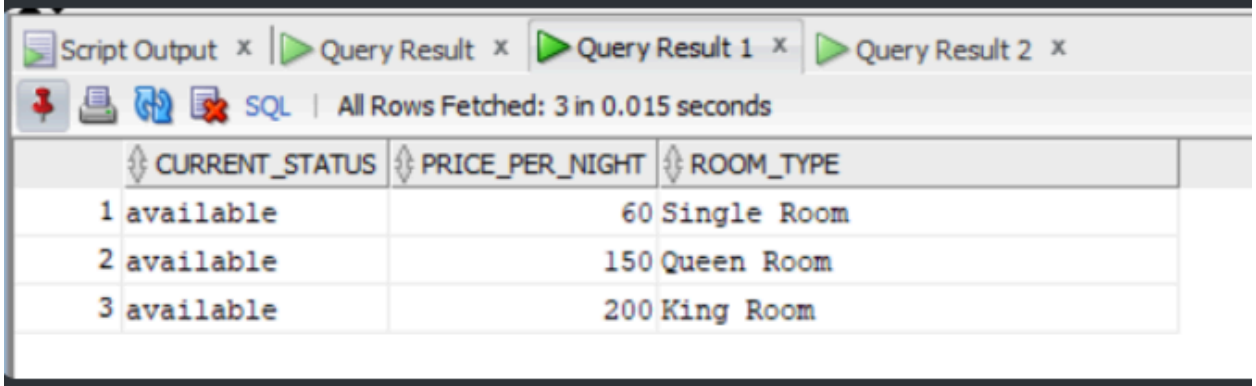
```
create view vwToDoSalesPromotion(first_name, last_name, email, total_points)
as
    select  Customer.first_name,
            Customer.last_name,
            Customer.email,
            Membership.total_points
    from Customer
    inner join Membership on Customer.customer_id=Membership.customer_id
    where Membership.total_points > 10000;


create view vwRoomsAvailability(current_status, room_num, price_per_night, room_type)
as
    (select status,
            room_num,
            price_per_night,
            room_type
    from Room
    where status='available');


create view vwEmployedLength(employee_id, first_name, position_, hiring_date)
as
    (select employee_id,
            first_name,
            position_,
            hiring_date
    from Employee
    where hiring_date < '2015-01-01');

select * from vwToDoSalesPromotion;
select * from vwRoomsAvailability;
select * from vwEmployedLength;
```



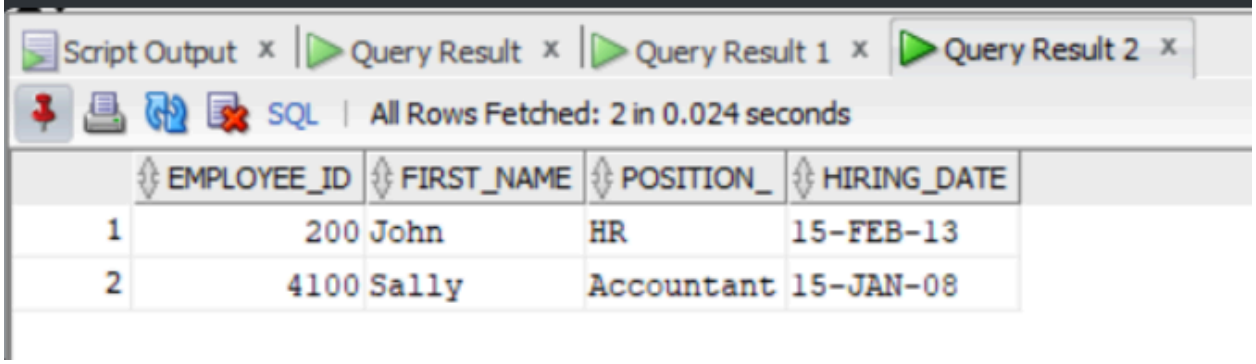
Outputs of the views



	FIRST_NAME	LAST_NAME	EMAIL	TOTAL_POINTS
1	John	Smith	thejohnsmith12@gmail.com	12345
2	Sally	Wrinkle	imoldl23@fossil.com	999999



	CURRENT_STATUS	PRICE_PER_NIGHT	ROOM_TYPE
1	available	60	Single Room
2	available	150	Queen Room
3	available	200	King Room



	EMPLOYEE_ID	FIRST_NAME	POSITION_	HIRING_DATE
1	200	John	HR	15-FEB-13
2	4100	Sally	Accountant	15-JAN-08

Simple Queries

```
-- Query that uses ORDER BY to get employees'salary in descending order.
select
    employee_id,
    last_name,
    position_,
    salaryPer_year as salary
from Employee
order by salary DESC;
```

Script Output | Query Result

SQL | All Rows Fetched: 4 in 0.018 seconds

	EMPLOYEE_ID	LAST_NAME	POSITION_	SALARY
1	1000	Water	Marketing manager	95000
2	4100	Deer	Accountant	76000
3	200	Marx	HR	70000
4	20040	Fox	Receptionist	45000

```
-- Query to find hotels' income that are greater than a specific number which is 5,000,000
select
    income,
    hotel_address
from Revenue
where income >= 5000000;
```

Script Output | Query Result

SQL | All Rows Fetched: 2 in 0.014 seconds

	INCOME	HOTEL_ADDRESS
1	10000000	510 Database Avenue, ON, M4C 2R5
2	8000000	110 Spring Avenue, ON, M4C 7R5

```sql
-- Query to find bills that exceed $10,000 in descending order
select
    bill_id,
    last_name,
    payment_date,
    amount
from Bill
where amount >= 10000 order by amount DESC;
```

Query Result

SQL | All Rows Fetched: 2 in 0.016 seconds

	BILL_ID	LAST_NAME	PAYMENT_DATE	AMOUNT
1	11	White	15-FEB-04	22568.52
2	7824	Andrews	15-JUL-23	10000.01

```sql
-- Query to find departments with 100 or more employees
select
    department_name,
    num_employees
from Department
where num_employees >= 100;
```

Query Result

SQL | All Rows Fetched: 1 in 0.016 seconds

	DEPARTMENT_NAME	NUM_EMPLOYEES
1	Service	163

```sql
-- Query to find available rooms
select
    room_num,
    bed_type,
    room_type,
    price_per_night
from Room
where status = 'available';
```

Query Result

SQL | All Rows Fetched: 2 in 0.028 seconds

	ROOM_NUM	BED_TYPE	ROOM_TYPE	PRICE_PER_NIGHT
1	2123	Queen	Queen Room	150
2	2250	King	King Room	200


```sql
-- Query to find available rooms
select
    room_num,
    bed_type,
    room_type,
    price_per_night
from Room
where status = 'available';
```

Query Result

SQL | All Rows Fetched: 2 in 0.028 seconds

	ROOM_NUM	BED_TYPE	ROOM_TYPE	PRICE_PER_NIGHT
1	2123	Queen	Queen Room	150
2	2250	King	King Room	200

Advanced queries

```
-- ======================
-- ADVANCED QUERIES BELOW
-- ======================

-- showing customers who have checked-in in the month of October 2024
select
    customer_id,
    first_name,
    last_name
from Customer
where customer_id
        in (select customer_id
        from Reservation
        where checkIN_date >= TO_DATE('2024-10-01','YYYY-MM-DD') and checkIN_date <= TO_DATE('2024-10-31','YYYY-MM-DD'));
```

Query Result

SQL | All Rows Fetched: 2 in 0.014 seconds

	CUSTOMER_ID	FIRST_NAME	LAST_NAME
1	S1111-22222-33333	Snake	Plisken
2	S1391-39139-13913	Eren	Jaeger

```
-- lists total revenue sorted by payment type
SELECT payment_type, SUM(amount) AS total_revenue
FROM Bill
GROUP BY payment_type
HAVING SUM(amount) > 0;
```

Query Result

SQL | All Rows Fetched: 3 in 0.022 seconds

	PAYMENT_TYPE	TOTAL_REVENUE
1	Debit	663.13
2	Cash	22568.52
3	Credit	11720.87

```sql
-- lists rooms that are available or occupied
SELECT room_num, bed_type, room_type, status
FROM Room
WHERE status = 'available'
UNION
SELECT room_num, bed_type, room_type, status
FROM Room
WHERE status = 'occupied';
```

uery Result

SQL | All Rows Fetched: 7 in 0.017 seconds

	ROOM_NUM	BED_TYPE	ROOM_TYPE	STATUS
1	1239	Double	Double Room	occupied
2	1510	Double	Single Room	occupied
3	2123	Queen	Queen Room	available
4	2250	King	King Room	available
5	3333	King	Deluxe Room	occupied
6	4001	Queen	Suite	occupied
7	4005	King	Presidential Suite	occupied

```sql
-- lists average price of each room type
SELECT room_type, AVG(price_per_night) AS avg_price
FROM Room
GROUP BY room_type
HAVING AVG(price_per_night) > 150;
```

uery Result

SQL | All Rows Fetched: 5 in 0.017 seconds

	ROOM_TYPE	AVG_PRICE
1	King Room	200
2	Presidential Suite	2000
3	Suite	1000
4	Studio	400
5	Deluxe Room	300

```sql
-- lists departments that have employees hired before 2014
SELECT department_id, department_name
FROM Department d
WHERE EXISTS (
    SELECT 1
    FROM Employee e
    WHERE e.department_id = d.department_id
      AND e.hiring_date < TO_DATE('2014-01-01', 'YYYY-MM-DD')
);
```

Query Result

SQL | All Rows Fetched: 2 in 0.023 seconds

	DEPARTMENT_ID	DEPARTMENT_NAME
1	2	HR
2	7	Accounting

UNIX shell

```
================================================================
| Oracle All Inclusive Tool                                    |
| Main Menu - Select Desired Operation(s):                     |
| <CTRL-Z Anytime to Enter Interactive CMD Prompt>             |
================================================================

  1) Drop Tables
  2) Create Tables
  3) Create Views
  4) Populate Tables
  5) Query Tables
  6) Query Views

  E) End/Exit
Choose:
```

Usage example:

```
============================================================
| Oracle All Inclusive Tool                                |
| Main Menu - Select Desired Operation(s):                 |
| <CTRL-Z Anytime to Enter Interactive CMD Prompt>          |
============================================================

  1) Drop Tables
  2) Create Tables
  3) Create Views
  4) Populate Tables
  5) Query Tables
  6) Query Views

  E) End/Exit
Choose:
1

SQL*Plus: Release 12.1.0.2.0 Production on Fri Nov 29 05:06:54 2024

Copyright (c) 1982, 2014, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SQL> SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.
```

Functional Dependency

FD for strong entities:

Department

{department_id} —> department_name, num_employees

Employee

{employee_id} —> last_name, first_name, position, hiring_date, salaryPer_year, SSN, department_id

Customer

{customer_id} —> first_name, last_name, gender, email, ssn, dob

Room

{room_num} —> bed_type, room_type, price_per_night, status, memo

Reservation

{reservation_id} —> customer_id , room_num , stay_period , checkIN_date , status, first_name ,last_name

FD for weak entities:

Bill

{reservation_id, bill_id} —> last_name, amount, payment_type, payment_date

Membership

{customer_id} —> first_name, last_name, start_date, total_points

3NF

Ex. 2: Table with a transitive FD to 3NF
Employee(emp_id, emp_name, dept_id, dept_name, location)

The above table schema has the following FDs:
emp_id → emp_name, dept_id
dept_id → dept_name, location

The FD in red is a **transitive dependency**, as dept_name and location are determined by dept_id, which is determined by emp_id, so dept_name and location are indirectly dependent on emp_id meaning this table violates 3NF. In order to solve this we must decompose the table into two separate tables: Employee and Department.

Employee(emp_id, emp_name, dept_id)
{emp_id} → {emp_name, dept_id}

Department(dept_id, dept_name, location}
{dept_id} → {dept_name, location}

Now both tables are in 2NF, and have no transitive dependencies, making them 3NF.

Hotel: Not In 3NF
Hotel(hotel_id, hotel_address)
The hotel table has a composite attribute hotel_address (violates 1NF) so it must be broken down into its components

Hotel(hotel_id, province, city, street_name, street_number, postal_code)
{hotel_id} → {province, city, street_name, street_number, postal_code}

The schema now satisfies 1NF, and has no partial or transitive dependencies meaning it is **in 3NF**

Revenue: Not In 3NF
Revenue(hotel_id, fiscal_year, fiscal_quarter, income, taxes, salary_paid,
maintenance_cost, hotel_address)
The revenue table has a partial dependency {hotel_id} -> {hotel_address} (violates 2NF) so in order to fix it we must remove hotel_address from the Revenue table:

Revenue(hotel_id, fiscal_year, fiscal_quarter, income, taxes, salary_paid,
maintenance_cost)
{hotel_id, fiscal_year, fiscal_quarter} → {income, taxes, salary_paid, maintenance_cost}

This schema now satisfies 2NF and has no transitive dependencies meaning it is in 3NF

Reservation: Not In 3NF
Reservation(res_id, customer_id, room_num, stay_period, check_in_date, status,
f_name, l_name)
The reservation table has a partial dependency {customer_id} -> {f_name, l_name} (violates 2NF) so in order to fix it we must remove f_name and l_name from the table:

Reservation(res_id, customer_id, room_num, stay_period, check_in_date, status)
{res_id} → {customer_id, room_num, stay_period, check_in_date, status}

This schema now satisfies 2NF and has no transitive dependencies meaning it is in 3NF

Bill: Not In 3NF
Bill(bill_id, l_name, amount, payment_type, payment_date)
To make into a 3NF, remove last_name, and replace by "customer_id", so that
The last_name can be accessed by "customer_id"

Bill(bill_id, customer_id, amount, payment_type, payment_date)
{bill_id} → {customer_id, amount, payment_type, payment _date}.

Room: In 3NF
Room(room_num, bed_type, room_type, price_per_night, status, memo)
{room_num} → {bed_type, room_type, price_per_night, status, memo}

Membership: Not In 3NF
Membership(customer_id, f_name, l_name, start_date, total_points)

The membership table has redundant attributes f_name and l_name (connected by customer_id) so we must remove them

Membership(customer_id, start_date, total_points)
{customer_id} → {start_date, total_points}

This schema now satisfies 2NF and has no transitive dependencies meaning it is in 3NF

BCNF And Bernstein Algorithm

Find one table that has partial dependency and transitive dependency, and then use both BCNF and bernstein algorithm to decompose the table.

Revenue(hotel_id, fiscal_year, fiscal_quarter, income, taxes, salary_paid, maintenance_cost, hotel_address, postal_code)

Transitive dependency:
Hotel_address -> postal_code

Partial dependency:
{hotel_id} -> hotel_address
{hotel_id} -> postal_code

Map:
hotel_id = A, fiscal_year = B, fiscal_quarter = C
income = D, taxes = E, salary_paid = F, maintenance_cost = G,
hotel_address = H, postal_code = I

Using Bernstein algorithm for Revenue table

1. FD={ABC -> DEFG,
   A -> HI,
   H -> I}
2. FD={ABC -> D,
   ABC -> E,
   ABC -> F,
   ABC -> G,
   A -> H,
   A -> I,
   H -> I}

ABC → D: $\{ABC\}^+=\{ABCEFGHI\}$	no D, not redundant
ABC → E: $\{ABC\}^+=\{ABCDFGHI\}$	no E, not redundant
ABC → F: $\{ABC\}^+=\{ABCDEGHI\}$	no F, not redundant
ABC → G: $\{ABC\}^+=\{ABCDEFHI\}$	no G, not redundant
A → H: $\{A\}^+=\{AI\}$	no H, not redundant
A → I: $\{A\}^+=\{AHI\}$	yes I, redundant
H → I: $\{H\}^+=\{H\}$	no I, not redundant

After removing redundancies: FD={ABC -> D, ABC -> E, ABC -> F, ABC -> G, A -> H, H -> I,}
Minimizing left side by removing partial dependencies:

ABC -> D	$\{A\}^+=\{AHI\}$	$\{B\}^+=\{B\}$	$\{C\}^+=\{C\}$	no D, FD is fully dependent
ABC -> E	$\{A\}^+=\{AHI\}$	$\{B\}^+=\{B\}$	$\{C\}^+=\{C\}$	no E, FD is fully dependent
ABC -> F	$\{A\}^+=\{AHI\}$	$\{B\}^+=\{B\}$	$\{C\}^+=\{C\}$	no F, FD is fully dependent

ABC -> G $\{A\}^+=\{AHI\}$ $\{B\}^+=\{B\}$ $\{C\}^+=\{C\}$ no G, FD is fully dependent

3. {DEFG} only appear on RHS so they are not part of keys
   {ABC} only appear on LHS so they are part of keys
   Thus, $\{ABC\}^+=\{ABCDEFGHI\}$
   $\{ABCH\}^+=\{ABCDEFGHI\}$
   $\{ABCI\}^+=\{ABCDEFGHI\}$
   all are keys
4. FD={ABC -> D, ABC -> E, ABC -> F, ABC -> G, A -> H, H -> I,}
   FD={ABC -> DEFG, A -> H, H -> I,}
   New tables
   R1=(ABCDEFG) R2=(AH) R3=(HI)

Using BCNF for Revenue table
R=(ABCDEFGHI)
FD={ABC -> DEFG,
A -> HI,
H -> I}
$\{ABC\}^+=\{ABCDEFGHI\}$
$\{A\}^+=\{AHI\}$
$\{H\}^+=\{HI\}$
Since R is not bcnf with respect to A -> HI and H -> I
Decompose into R1 and R2, where
R1=(ABCDEFG)is bcnf
R2=(AHI) is bcnf

Thus, bcnf schema of R is
R1=(ABCDEFG)
R2=(AHI)

JAVA UI

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.util.Scanner;

public class JdbcOracleConnectionTemplate {

    public static void main(String[] args) {
        String dbURL = "jdbc:oracle:thin:nsiddele/02156392@oracle.scs.ryerson.ca:1521:orcl";
        String response = "a";
        Scanner in = new Scanner(System.in);

        while (response != null) {
            System.out.println("==================================");
            System.out.println("Welcome to the database manager");
            System.out.println("");
            System.out.println("( 1 ) > Drop Tables");
            System.out.println("( 2 ) > Create Tables");
            System.out.println("( 3 ) > Populate Tables");
            System.out.println("( 4 ) > Query Tables");
            System.out.println("( 5 ) > Update Tables");
            System.out.println("( 6 ) > Alter Tables");
            System.out.println("");
            System.out.println("( E ) > Exit");
            System.out.println("");
            System.out.println("==================================");
            System.out.println("");
            System.out.println("Enter option > ");

            response = in.nextLine();

            if(response.equals("1")) {
                dropTables(dbURL);
            }else if(response.equals("2")) {
                createTables(dbURL);
            }else if(response.equals("3")) {
                populateTables(dbURL);
            }else if(response.equals("4")) {
                customQuery(dbURL);
            }else if(response.equals("5")) {
                customUpdate(dbURL);
            }else if(response.equals("6")) {
                customAlter(dbURL);
            }else if(response.toUpperCase().equals("E")) {
                in.close();
                System.exit(0);
            }else {
                System.out.println("Please enter a valid option");
            }
        }
```

```java
    private static void dropTables(String dbURL) {

        String[] statements = {
                "drop table room cascade constraints",
                "drop table revenue cascade constraints",
                "drop table reservation cascade constraints",
                "drop table membership cascade constraints",
                "drop table hotel cascade constraints",
                "drop table employee cascade constraints",
                "drop table department cascade constraints",
                "drop table customer cascade constraints",
                "drop table bill cascade constraints"
        };

        for(String drop : statements) {

            try (Connection conn = DriverManager.getConnection(dbURL);
                 Statement stmt = conn.createStatement();)
            {

                stmt.executeUpdate(drop);
                System.out.println("Table dropped.");
                conn.close();

            } catch (SQLException e) {
                System.out.println("error: ");
                e.printStackTrace();
            }

        }

    }
```

```
90      private static void createTables(String dbURL) {
91
92          String[] statements = {
93                  "create table Hotel ("
```

```
167         };
168
169         for(String create : statements) {
170
171             try (Connection conn = DriverManager.getConnection(dbURL);
172                  Statement stmt = conn.createStatement();)
173             {
174
175                 stmt.executeUpdate(create);
176                 System.out.println("Table Created.");
177                 conn.close();
178
179             } catch (SQLException e) {
180                 System.out.println("error: ");
181                 e.printStackTrace();
182             }
183
184         }
185     }
```

```
187     private static void populateTables(String dbURL) {
188
189         String[] statements = {
190                 "INSERT INTO HOTEL (hotel_id, province, c
```

```
242                 "INSERT INTO Membership (customer_id, start_date, total_po
243         };
244
245         for(String row : statements) {
246
247             try (Connection conn = DriverManager.getConnection(dbURL);
248                  Statement stmt = conn.createStatement();)
249             {
250
251                 stmt.executeUpdate(row);
252                 System.out.println("Row inserted.");
253                 conn.close();
254
255             } catch (SQLException e) {
256                 System.out.println("error: ");
257                 e.printStackTrace();
258             }
259
260         }
261     }
```

```java
    private static void customQuery(String dbURL) {

        Scanner scan = new Scanner(System.in);
        System.out.println("Enter a query with proper SQL syntax >  ");
        String query = scan.nextLine();

        try (Connection conn = DriverManager.getConnection(dbURL);
             Statement stmt = conn.createStatement();)
        {

            ResultSet rs = stmt.executeQuery(query);
            ResultSetMetaData rsmd = rs.getMetaData();

            int numCols = rsmd.getColumnCount();

            for(int i=1; i<=numCols; i++) {
                String colName = rsmd.getColumnName(i);
                System.out.printf("%-25s", colName);
            }
            System.out.print("\n");

            while(rs.next()) {
                for(int i=1; i<=numCols; i++) {
                    System.out.printf("%-25s", rs.getObject(i).toString());
                }
                System.out.print("\n");
            }


            conn.close();

        } catch (SQLException e) {
            System.out.println("error: ");
            e.printStackTrace();
        }
    }

```

```java
    private static void customUpdate(String dbURL) {

        Scanner scan = new Scanner(System.in);
        System.out.println("Enter one of the following statements (INSERT, UPDATE, DELETE) with proper SQL syntax >  ");
        String update = scan.nextLine();

        try (Connection conn = DriverManager.getConnection(dbURL);
             Statement stmt = conn.createStatement();)
        {

            int rowsAff = stmt.executeUpdate(update);
            System.out.println(String.format("%d rows affected", rowsAff));

            conn.close();

        } catch (SQLException e) {
            System.out.println("error: ");
            e.printStackTrace();
        }
    }

```

```java
    private static void customAlter(String dbURL) {

        Scanner scan = new Scanner(System.in);
        System.out.println("Enter an ALTER statement with proper SQL syntax >  ");
        String update = scan.nextLine();

        try (Connection conn = DriverManager.getConnection(dbURL);
             Statement stmt = conn.createStatement();)
        {

            int rowsAff = stmt.executeUpdate(update);
            System.out.println("Column altered");
            System.out.println(String.format("%d rows affected", rowsAff));

            conn.close();

        } catch (SQLException e) {
            System.out.println("error: ");
            e.printStackTrace();
        }
    }
}
```

Example output:

```
====================================
Welcome to the database manager

( 1 ) > Drop Tables
( 2 ) > Create Tables
( 3 ) > Populate Tables
( 4 ) > Query Tables
( 5 ) > Update Tables
( 6 ) > Alter Tables

( E ) > Exit

====================================

Enter option >
4
Enter a query with proper SQL syntax >
select * from employee
EMPLOYEE_ID              LNAME                    FNAME                    HIRE_DATE                POSITION_                SSN                      SALARY                   DEPARTMENT_ID
1000                     Water                    Alice                    2015-01-15 00:00:00.0    Marketing manager        123456780                95000                    1
200                      Marx                     John                     2012-02-15 00:00:00.0    HR                       546127780                70000                    2
4100                     Deer                     Sally                    2008-01-15 00:00:00.0    Accountant               123445666                76000                    7
20040                    Fox                      Ben                      2019-01-15 00:00:00.0    Receptionist             547789204                45000                    8
20041                    Bobson                   Bob                      2019-01-15 00:00:00.0    Janitor                  547789205                35000                    8
```

Relational Algebra For Queries

select department_name, num_employees from Department where num_employees >= 100;	$\sigma_{\text{num}_{\text{employees}} \geq 100}(\text{Department})$

select first_name, last_name, stay_period, status from Reservation where status = 'checked-in';	$\sigma_{\text{status} = \text{'checked-in'}}(\text{Reservation})$

SELECT room_num, bed_type, room_type, status FROM Room WHERE status = 'available' UNION SELECT room_num, bed_type, room_type, status FROM Room WHERE status = 'occupied';	$\pi_{\text{room}_{\text{num}},\text{bed}_{\text{type}},\text{room}_{\text{type}},\text{status}}\left(\sigma_{\text{status} = \text{'available'}}(\text{Room})\right)$ U $\pi_{\text{room}_{\text{num}},\text{bed}_{\text{type}},\text{room}_{\text{type}},\text{status}}\left(\sigma_{\text{status} = \text{'occupied'}}(\text{Room})\right)$

```
SELECT customer_id, first_name, last_name
FROM Customer
MINUS
SELECT customer_id, first_name, last_name
FROM Membership;
```

$$\pi_{\text{customer\_id,first\_name,last\_name}}(Customer) - \pi_{\text{customer\_id,first\_name,last\_name}}(Membership)$$

```
select
   first_name,
   last_name,
   total_points
from Membership
where total_points >= 10000 order by total_points DESC;
```

$$\tau_{total\ points}\ DESC\ \boldsymbol{\pi}_{first\ name,\ last\ name,\ total\ points}(\boldsymbol{\sigma}_{total\ points>=10000}(Membership))$$