Oskar Bartoszyński projekt bazy danych hotelu w firebird

DIAGRAM CHEN’A

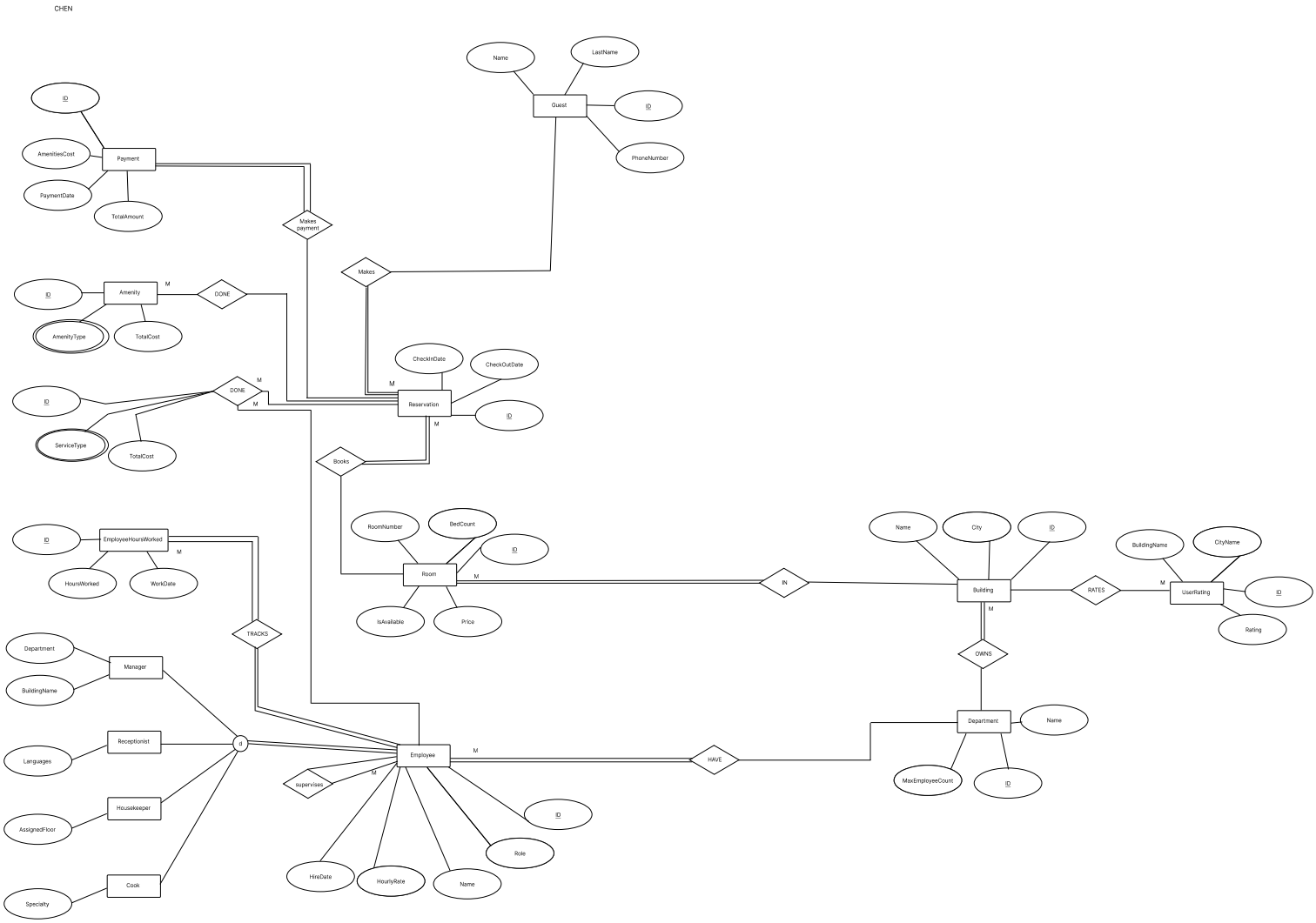


DIAGRAM UML

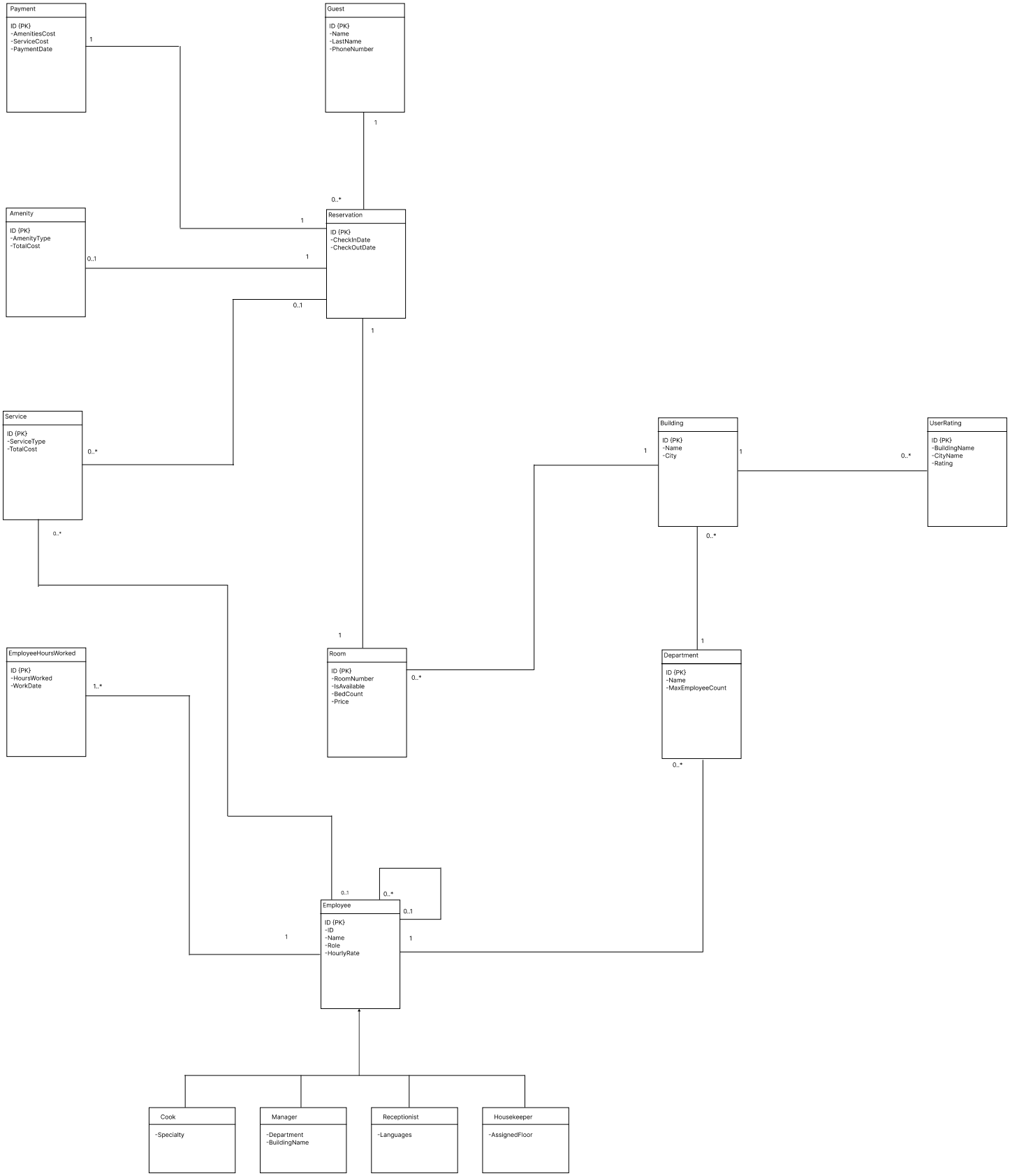

Payment
ID {PK}
-AmenitiesCost
-ServiceCost
-PaymentDate
Guest
ID {PK}
-Name
-LastName
-PhoneNumber
Amenity
ID {PK}
-AmenityType
-TotalCost
Reservation
ID {PK}
-CheckInDate
-CheckOutDate
Service
ID {PK}
-ServiceType
-TotalCost
Building
ID {PK}
-Name
-City
UserRating
ID {PK}
-BuildingName
-CityName
-Rating
EmployeeHoursWorked
ID {PK}
-HoursWorked
-WorkDate
Room
ID {PK}
-RoomNumber
-IsAvailable
-BedCount
-Price
Department
ID {PK}
-Name
-MaxEmployeeCount
Employee
ID {PK}
-ID
-Name
-Role
-HourlyRate
Cook
-Specialty
Manager
-Department
-BuildingName
Receptionist
-Languages
Housekeeper
-AssignedFloor
1
0..*
0..1
1..*

DIAGRAM BARKER’A

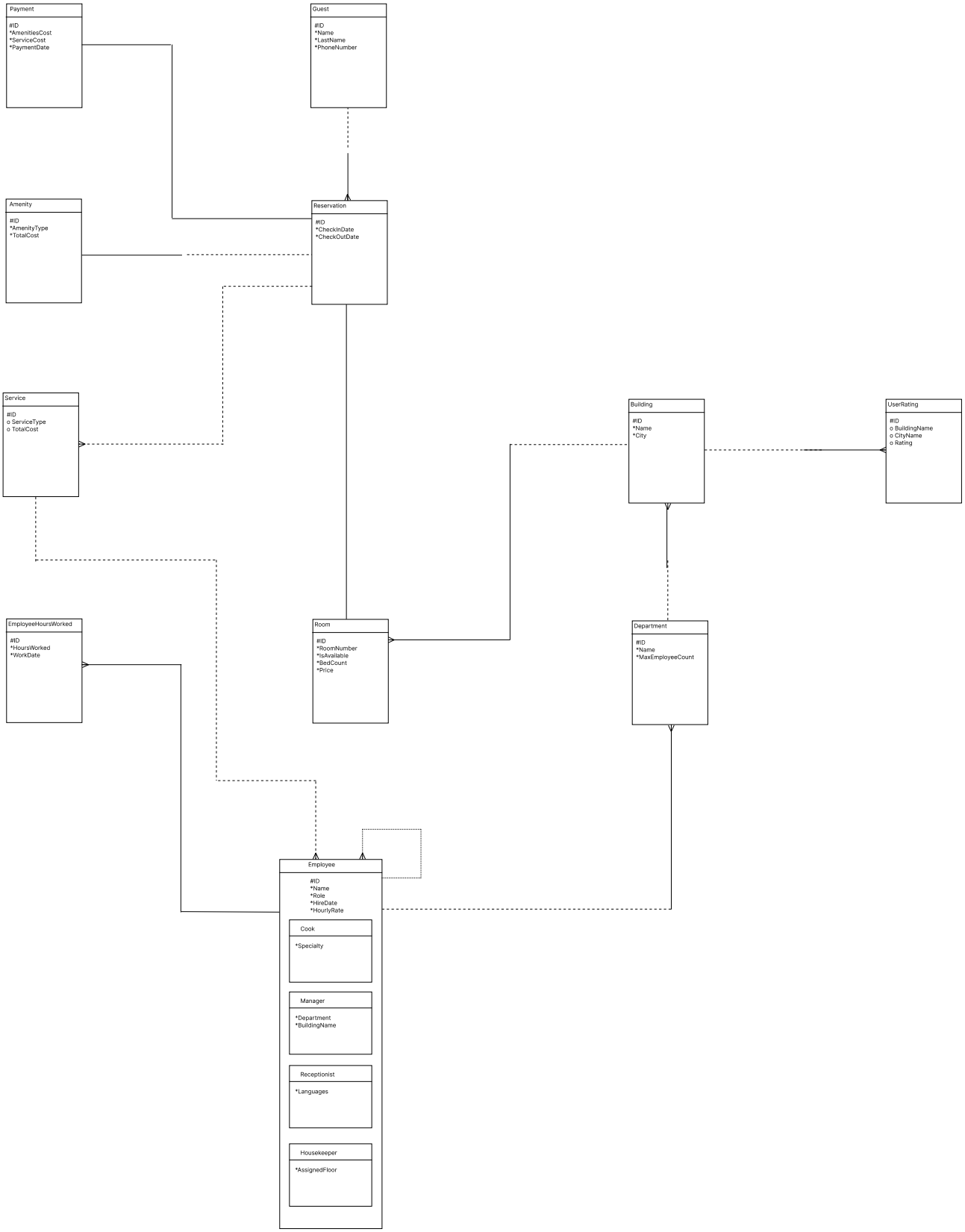
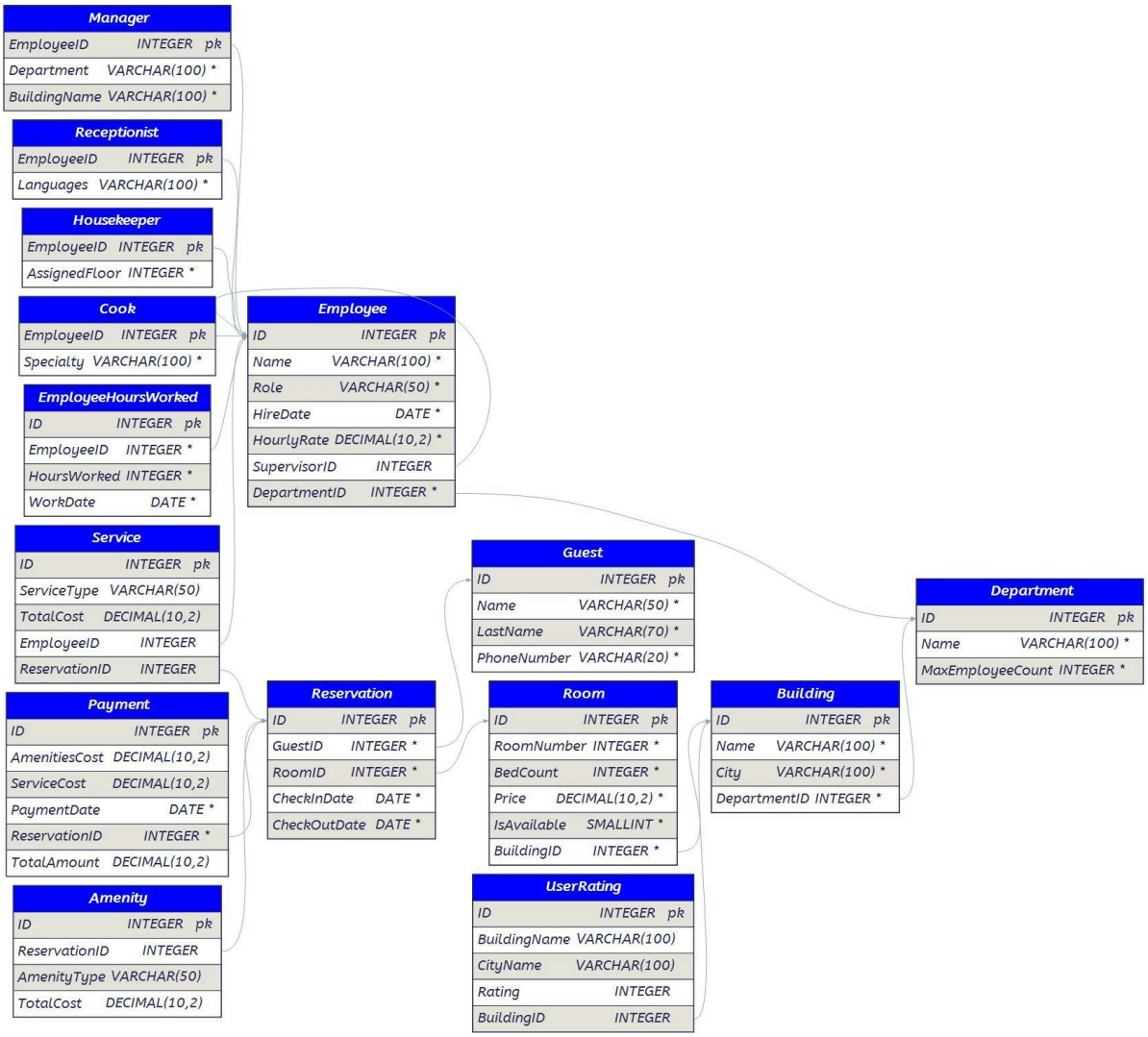


DIAGRAM RELACYJNY



Inner join

```
SELECT
    Room.RoomNumber,
    Room.BedCount,
    Building.Name AS BuildingName
FROM
    Room
INNER JOIN
    Building
ON
    Room.BuildingID = Building.ID;
```

ROOMNUMBER	BEDCOUNT	BUILDINGNAME
111	2	Building A
112	3	Building A
113	1	Building B
114	4	Building B
115	2	Building C
116	3	Building C
117	5	Building D
118	6	Building D
119	2	Building B
120	3	Building B
121	4	Building C
122	1	Building C
123	2	Building A
124	5	Building A
125	1	Building B
126	4	Building B
127	3	Building C
128	2	Building C
129	5	Building D
130	2	Building D

ROOMNUMBER	BEDCOUNT	BUILDINGNAME
131	3	Building E
132	6	Building E
133	2	Building A
134	3	Building A
135	1	Building B
136	4	Building B
137	3	Building C
138	2	Building C

Outer join

```
SELECT
   E1.Name AS EmployeeName,
   E2.Name AS SupervisorName
FROM
   Employee E1
LEFT JOIN
   Employee E2
ON
   E1.SupervisorID = E2.ID;
```

```
EMPLOYEENAME                                                                      SUPERVISORNAME
================================================================================ ==============================================
Jan Kowalski                                                                       <null>
Anna Nowak                                                                         Jan Kowalski
Piotr Wiśniewski                                                                  Anna Nowak
Maria Dąbrowska                                                                   Jan Kowalski
Tomasz Lewandowski                                                                 <null>
Katarzyna Zielińska                                                               Tomasz Lewandowski
Michał Szymański                                                                 Piotr Wiśniewski
Agnieszka Kozłowska                                                               Tomasz Lewandowski
Robert Jankowski                                                                   <null>
Barbara Wojciechowska                                                              Robert Jankowski
Adam Mazur                                                                         Agnieszka Kozłowska
Ewa Kwiatkowska                                                                    Robert Jankowski
Krzysztof Pawlak                                                                   <null>
Monika Górska                                                                     Krzysztof Pawlak
Paweł Dudek                                                                       Ewa Kwiatkowska
Dorota Grabowska                                                                   Krzysztof Pawlak
Marcin Nowicki                                                                     <null>
Aleksandra Adamczyk                                                                Marcin Nowicki
Rafał Kowalczyk                                                                   Dorota Grabowska
Magdalena Wójcik                                                                  Marcin Nowicki
```

grupowanie

```
SELECT
    Building.Name AS BuildingName,
    COUNT(Room.ID) AS RoomCount
FROM
    Room
INNER JOIN
    Building
ON
    Room.BuildingID = Building.ID
GROUP BY
    Building.Name;
```

BUILDINGNAME	ROOMCOUNT
Building A	6
Building B	8
Building C	8
Building D	4
Building E	2

Sortowanie

```
SELECT
    Name,
    LastName
FROM
    Guest
ORDER BY
   Name ASC;
```

NAME	LASTNAME
Adam	Marchewka
Alicja	Olszewska
Ewa	Trombalska
Stuś	Pędziwiatr
Stuś	Pędziwiatr
Tomek	Pietruszka

podzapytanie skorelowane

```
SELECT
   RoomNumber,
   Price
FROM
   Room R1
WHERE
   Price < (
      SELECT
         AVG(R2.Price)
      FROM
         Room R2
      WHERE
         R2.BuildingID = R1.BuildingID
   );
```

ROOMNUMBER	PRICE
111	160.00
113	95.00
115	150.00
117	290.00
119	175.00
122	110.00
123	160.00
125	95.00
128	145.00
130	160.00
131	190.00
133	140.00
135	85.00
138	170.00

podzapytanie nieskorelowane

```
SELECT
   RoomNumber,
   Price
FROM
   Room
WHERE
   Price > (
      SELECT
         AVG(Price)
      FROM
         Room
   );
```

ROOMNUMBER	PRICE
112	210.00
114	240.00
117	290.00
118	350.00
120	220.00
121	270.00
124	320.00
126	250.00
129	305.00
132	350.00
136	230.00

having

```
SELECT
   BuildingID,
   AVG(Price) AS AveragePrice
FROM
   Room
GROUP BY
   BuildingID
HAVING
   AVG(Price) > 200;
```

BUILDINGID	AVERAGEPRICE
4	320.00
8	240.00
11	232.50
12	270.00

in

```
SELECT
   Name,
   LastName,
   PhoneNumber
FROM
   Guest
WHERE
   PhoneNumber IN ('111222333', '987654321','542986742','490653450');
```

NAME	LASTNAME	PHONENUMBER
Tomek	Pietruszka	111222333

Any

ROOMNUMBER	BEDCOUNT
112	3
114	4
116	3
117	5
118	6
120	3
121	4
124	5
126	4
127	3
129	5
131	3
132	6
134	3
136	4
137	3

```
SELECT
   RoomNumber,
   BedCount
FROM
   Room
WHERE
   BedCount > ANY (
      SELECT
         BedCount
      FROM
         Room R
         INNER JOIN Building B ON R.BuildingID = B.ID
      WHERE
         B.Name = 'Building A'
   );
```

All

ROOMNUMBER	PRICE
117	290.00
118	350.00
121	270.00
124	320.00
129	305.00
132	350.00

```
SELECT
   RoomNumber,
   Price
FROM
   Room
WHERE
   Price > ALL (
      SELECT
         Price
      FROM
         Room R
         INNER JOIN Building B ON R.BuildingID = B.ID
      WHERE
         B.Name = 'Building B'
   );
```

like

```
SELECT
    Name,
    LastName
FROM
    Guest
WHERE
    Name LIKE 'A%';
```

NAME	LASTNAME
Adam	Marchewka
Alicja	Olszewska

Exists

```
SELECT
    RoomNumber,
    Price
FROM
    Room R
WHERE
    EXISTS (
        SELECT
            1
        FROM
            Building B
        INNER JOIN Department D ON B.DepartmentID = D.ID
        WHERE
            R.BuildingID = B.ID
            AND D.Name = 'Housekeeping'
    );
```

ROOMNUMBER	PRICE
113	95.00
114	240.00
123	160.00
124	320.00
131	190.00
132	350.00

==========PROCEDURA==========

```
SET TERM ^ ;

CREATE OR ALTER PROCEDURE MarkRoomAsUnavailable (
    RoomIDInput INTEGER
)
AS
BEGIN
    UPDATE Room
    SET IsAvailable = 0
    WHERE ID = :RoomIDInput;
END^

SET TERM ; ^
```

==========PROCEDURA==========

```
SET TERM ^ ;
CREATE OR ALTER PROCEDURE MarkRoomAsAvailable (
    RoomIDInput INTEGER
)
AS
DECLARE VARIABLE CurrentDate DATE;
BEGIN
    -- Pobierz aktualną datę
    CurrentDate = CURRENT_DATE;

    -- Zmień status pokoju na dostępny, jeśli rezerwacja zakończyła się
    UPDATE Room
    SET IsAvailable = 1
    WHERE ID = :RoomIDInput
      AND NOT EXISTS (
          SELECT 1
          FROM Reservation
          WHERE RoomID = :RoomIDInput
            AND CheckOutDate >= :CurrentDate
      );
END^
SET TERM ; ^
```

==========WYZWALACZ==========

```
SET TERM ^ ;
CREATE OR ALTER TRIGGER AfterInsertReservation
AFTER INSERT ON Reservation
AS
BEGIN
   EXECUTE PROCEDURE MarkRoomAsUnavailable(NEW.RoomID);
END^

SET TERM ; ^
```

==========WYZWALACZ==========

```
SET TERM ^ ;
CREATE OR ALTER TRIGGER AfterUpdateReservation
AFTER UPDATE OF CheckOutDate ON Reservation
AS
BEGIN
   EXECUTE PROCEDURE MarkRoomAsAvailable(NEW.RoomID);
END^

SET TERM ; ^
```

==========WYZWALACZ==========

```
SET TERM ^ ;

CREATE OR ALTER TRIGGER RoomStatusCheckAfterUpdate
AFTER UPDATE ON Reservation
AS
DECLARE VARIABLE RoomID INTEGER;
BEGIN
   FOR SELECT ID
      FROM Room
      WHERE IsAvailable = 0
        AND NOT EXISTS (
            SELECT 1
            FROM Reservation
            WHERE RoomID = Room.ID
              AND CheckOutDate >= CURRENT_DATE
        )
   INTO :RoomID DO
   BEGIN
      EXECUTE PROCEDURE MarkRoomAsAvailable(:RoomID);
```

```
    END
END^

SET TERM ; ^
```

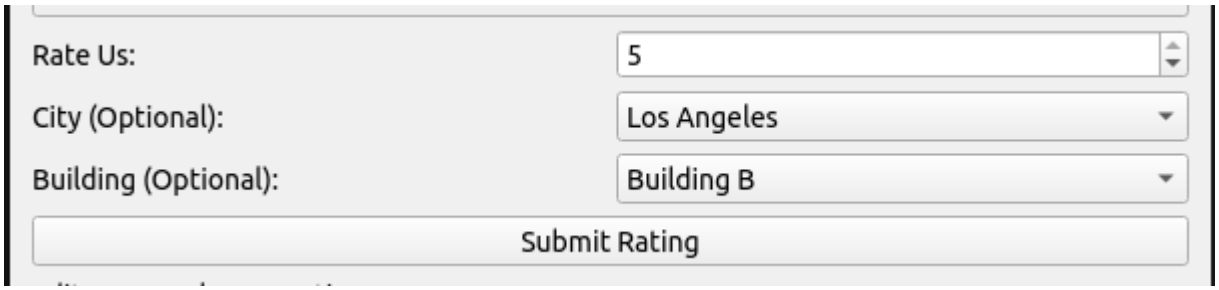
Raport ze zmniejszeniem zakresu EMPLOYEEID (do 10 z 20) oraz posortowanie malejąco po ilości godzin przepracowanych

ID	EMPLOYEEID	HOURSWORKED	WORKDATE
1	1	8	2024-12-01
4	4	8	2024-12-01
8	8	8	2024-12-01
5	5	8	2024-12-01
2	2	7	2024-12-01
6	6	7	2024-12-01
9	9	7	2024-12-01
3	3	6	2024-12-01
7	7	6	2024-12-01
10	10	6	2024-12-01

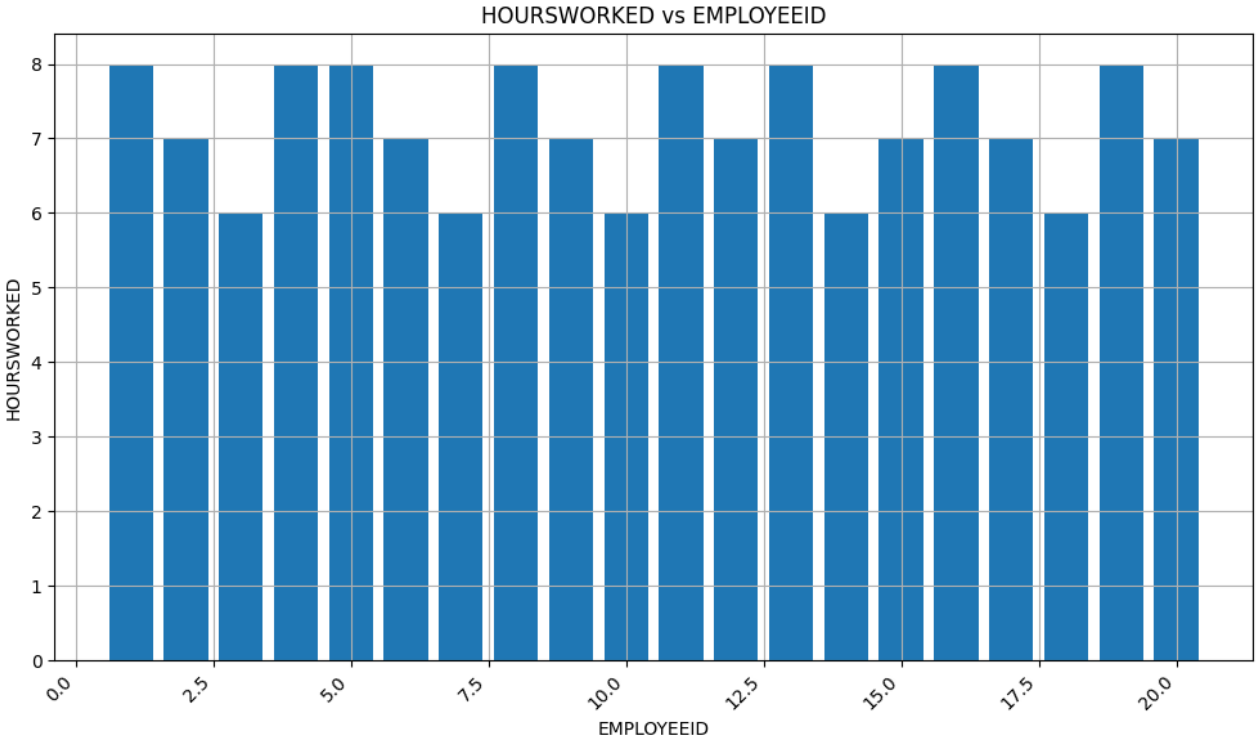
Raport z grupowaniem po SERVICETYPE oraz ustawieniem braku ID

	SERVICETYPE	TOTALCOST	EMPLOYEEID	RESERVATIONID
1	Birthday ...	400	nan	6
2	Child Care ...	250	nan	3
3	Child Care ...	250	nan	6
4	Cleaning	100	nan	1
5	Cleaning	100	nan	2
6	Cleaning	100	nan	3
7	Cleaning	100	nan	4
8	Cleaning	100	nan	6
9	Food Service	150	nan	1
10	Food Service	150	nan	2
11	Food Service	150	nan	3
12	Food Service	150	nan	4
13	Food Service	150	nan	6
14	Housekeeping	50	nan	1
15	Housekeeping	50	nan	4
16	Housekeeping	50	nan	6

Raport w formie formularza



Wykres przepracowanych godzin kpracowników (x – pracownicy(id); y -ilość godzin pracy)



Opis integracji aplikacji do raportowania z aplikacją kleincką

```
if __name__ == "__main__":
    import sys
    from raportgui import DatabaseApp  ← importuje main funckję (inicjalizującą działanie
                                          programu) z aplikacji raportującej do
                                          aplikacji klienckiej

    app = QApplication(sys.argv)


    window1 = HotelReservationApp()
    window1.resize(600, 800)
    window1.show()

    window2 = DatabaseApp()
    window2.resize(1000, 800)
    window2.show()


    sys.exit(app.exec_())
```

Obie aplikację połączone są bezpośrednio do bazy dancyh przy pomocy bibliotek wspierających połączenie z firebird.

```
from firebird.driver import connect, driver_config
```

Konfiguracja połączenia z bazą Firebird

```
driver_config.server_defaults.host.value = 'localhost'
DATABASE = '/opt/firebird/hotel.fdb'
USER = 'SYSDBA'
PASSWORD = 'SYSDBA'
```

Funkcja łącząca się do bazy

```
def connect_to_db():
    return connect(database=DATABASE, user=USER, password=PASSWORD)
```

oraz stworzenie tzw kursora

```
self.cur = self.con.cursor()
```

kluczowe fragmenty kodu źródłowego

```
def load_details(self):
    try:
        cursor = self.conn.cursor()
        cursor.execute("""
            SELECT g.PhoneNumber, r.CheckInDate, r.CheckOutDate
            FROM Reservation r
            JOIN Guest g ON r.GuestID = g.ID
            WHERE r.ID = ?
        """, (self.reservation_id,))
        details = cursor.fetchone()
        if not details:
            show_error("Could not retrieve reservation details.")
            self.reject()
            return


def search_rooms(self):
    try:
        cursor = self.conn.cursor()
        query = """
        SELECT r.RoomNumber, r.Price FROM Room r
        JOIN Building b ON r.BuildingID = b.ID
        WHERE r.BedCount = ? AND b.City = ? AND r.IsAvailable = 1
        """
        cursor.execute(query, (self.bed_count_spin.value(), self.city_combo.currentText()))
        rooms = cursor.fetchall()
        self.rooms_table.setRowCount(0)
        for room in rooms:
            row_pos = self.rooms_table.rowCount()
            self.rooms_table.insertRow(row_pos)
            self.rooms_table.setItem(row_pos, 0, QTableWidgetItem(str(room[0])))
            self.rooms_table.setItem(row_pos, 1, QTableWidgetItem(str(room[1])))
    except Exception as e:
        show_error(f"Could not fetch rooms: {e}")


try:
        cursor = self.conn.cursor()
        cursor.execute("""
            SELECT r.ID, g.Name, g.LastName, ro.RoomNumber, r.CheckInDate, r.CheckOutDate
            FROM Reservation r
            JOIN Guest g ON r.GuestID = g.ID
            JOIN Room ro ON r.RoomID = ro.ID
            WHERE g.PhoneNumber = ?
        """, (phone_number,))
```

```
        reservations = cursor.fetchall()
        self.reservation_table.setRowCount(0)
        if not reservations:
            show_info("No reservations found for the provided phone number.")
            return
```

Rezygnacja rezerwacji:

```
def cancel_reservation(self):
    selected_items = self.reservation_table.selectedItems()
    if not selected_items:
        show_warning("Please select a reservation to cancel.")
        return
    reservation_id = selected_items[0].text()

    try:
        cursor = self.conn.cursor()
        # Zakładamy, że kluczem jest ID (Reservation.ID)
        cursor.execute("DELETE FROM Reservation WHERE ID = ?", (reservation_id,))
```

Dodawanie rezerwacji:

```
 cursor.execute("""
            INSERT INTO Guest (Name, LastName, PhoneNumber)
            VALUES (?, ?, ?)
            RETURNING ID
        """, (self.name_edit.text().strip(), self.last_name_edit.text().strip(),
self.phone_edit.text().strip()))
        guest_id = cursor.fetchone()[0]

        # Pobranie ID pokoju
        cursor.execute("SELECT ID FROM Room WHERE RoomNumber = ?", (room_number,))
        room_id = cursor.fetchone()[0]

        # Dodanie rezerwacji
        cursor.execute("""
            INSERT INTO Reservation (GuestID, RoomID, CheckInDate, CheckOutDate)
            VALUES (?, ?, ?, ?)
            RETURNING ID
        """, (guest_id, room_id, check_in_date.strftime("%Y-%m-%d"),
check_out_date.strftime("%Y-%m-%d")))
        reservation_id = cursor.fetchone()[0]

        # Dodanie udogodnień
        for cb in self.amenities_checkboxes:
            if cb.isChecked():
                cursor.execute("""
                    INSERT INTO Amenity (ReservationID, AmenityType, TotalCost)
                    VALUES (?, ?, ?)
                """, (reservation_id, cb.text(), self.amenity_costs[cb.text()]))
```

```
        # Dodanie usług
        for cb in self.services_checkboxes:
            if cb.isChecked():
                cursor.execute("""
                    INSERT INTO Service (ReservationID, ServiceType, TotalCost)
                    VALUES (?, ?, ?)
                """, (reservation_id, cb.text(), self.service_costs[cb.text()]))

        # Obliczenie całkowitego kosztu
        days = (check_out_date - check_in_date).days
        amenities_cost = sum(self.amenity_costs[a.text()] for a in self.amenities_checkboxes if
a.isChecked())
        services_cost = sum(self.service_costs[s.text()] for s in self.services_checkboxes if
s.isChecked())
        total_cost = (room_price * days) + amenities_cost + services_cost

        # Dodanie płatności
        payment_date = datetime.now().strftime("%Y-%m-%d")
        cursor.execute("""
            INSERT INTO Payment (ReservationID, AmenitiesCost, ServiceCost, TotalAmount,
PaymentDate)
            VALUES (?, ?, ?, ?, ?)
        """, (reservation_id, amenities_cost, services_cost, total_cost, payment_date))

        self.conn.commit()
        show_success(f"Room {room_number} booked successfully!\nTotal Cost: ${total_cost:.2f}\
nPayment Date: {payment_date}")

    except Exception as e:
        self.conn.rollback()
        show_error(f"Could not complete booking: {e}")
```

Zaktualizowanie rezerwacji:

```
cursor.execute("UPDATE Guest SET PhoneNumber = ? WHERE ID = ?", (phone, guest_id))
        # Zaktualizuj daty rezerwacji
        cursor.execute("UPDATE Reservation SET CheckInDate = ?, CheckOutDate = ? WHERE
ID = ?",
                       (check_in_date.strftime("%Y-%m-%d"), check_out_date.strftime("%Y-%m-%d"),
self.reservation_id))
        self.conn.commit()
```

Rating:

```
try:
        cursor = self.conn.cursor()
        cursor.execute("""
            INSERT INTO UserRating (BuildingName, CityName, Rating)
            VALUES (?, ?, ?)
        """, (building, city, rating))
        self.conn.commit()
```

AMENITY

	ID	RESERVATIONID	AMENITYTYPE	TOTALCOST
1	1	1	Swimming Pool	300
2	2	1	Fitness Center	200
3	3	3	Swimming Pool	300
4	4	6	Swimming Pool	300

BUILDING

ID	NAME	CITY	DEPARTMENTID
1	Building A	Los Angeles	1
2	Building B	Los Angeles	2
3	Building C	Los Angeles	3
4	Building D	Los Angeles	4
5	Building A	Manhattan	5
6	Building B	Manhattan	6
7	Building C	Manhattan	7
8	Building A	Bydgoszcz	8
9	Building B	Bydgoszcz	9
10	Building C	Bydgoszcz	10
11	Building D	Bydgoszcz	11
12	Building E	Bydgoszcz	12
13	Building A	Brooklyn	13
14	Building B	Brooklyn	14
15	Building C	Brooklyn	15

COOK

EMPLOYEEID	SPECIALTY
4	Sea cuisine
8	Baking
12	Sushi
16	Sea cuisine
20	Baking

DEPARTMENT

ID	NAME	MAXEMPLOYEECOUNT
1	IT	15
2	Housekeeping	50
3	Food & Beverage	40
4	Maintenance	25
5	Housekeeping	45
6	Food & Beverage	30
7	Maintenance	15
8	Housekeeping	20
9	Food & Beverage	40
10	Maintenance	25
11	Human Resources	15
12	Housekeeping	60
13	Food & Beverage	30
14	Maintenance	10
15	Human Resources	30

EMPLOYEE

ID	NAME	ROLE	HIREDATE	HOURLYRATE	SUPERVISORID	DEPARTMENTID
1	Jan Kowalski	Manager	2020-01-15	50	[NULL]	1
2	Anna Nowak	Receptionist	2021-03-20	25	1	2
3	Piotr Wiśniewski	Housekeeper	2022-05-10	20	2	3
4	Maria Dąbrowska	Cook	2021-07-05	30	1	4
5	Tomasz Lewandowski	Manager	2019-11-30	55	[NULL]	5
6	Katarzyna Zielińska	Receptionist	2022-02-14	26	5	1
7	Michał Szymański	Housekeeper	2023-04-22	21	3	2
8	Agnieszka Kozłowska	Cook	2020-09-18	32	5	3
9	Robert Jankowski	Manager	2018-06-25	60	[NULL]	4
10	Barbara Wojciechowska	Receptionist	2021-12-05	27	9	5
11	Adam Mazur	Housekeeper	2022-08-17	22	8	1
12	Ewa Kwiatkowska	Cook	2021-01-30	33	9	2
13	Krzysztof Pawlak	Manager	2019-04-12	58	[NULL]	3
14	Monika Górska	Receptionist	2022-10-08	28	13	4
15	Paweł Dudek	Housekeeper	2023-01-25	23	12	5
16	Dorota Grabowska	Cook	2020-05-14	34	13	1
17	Marcin Nowicki	Manager	2018-12-03	62	[NULL]	2
18	Aleksandra Adamczyk	Receptionist	2021-09-22	29	17	3
19	Rafał Kowalczyk	Housekeeper	2022-06-30	24	16	4
20	Magdalena Wójcik	Cook	2021-03-07	35	17	5

EMPLOYEE HOURS WORKED

ID	EMPLOYEEID	HOURSWORKED	WORKDATE
1	1	8	2024-12-01
2	2	7	2024-12-01
3	3	6	2024-12-01
4	4	8	2024-12-01
5	5	8	2024-12-01
6	6	7	2024-12-01
7	7	6	2024-12-01
8	8	8	2024-12-01
9	9	7	2024-12-01
10	10	6	2024-12-01
11	11	8	2024-12-01
12	12	7	2024-12-01
13	13	8	2024-12-01
14	14	6	2024-12-01
15	15	7	2024-12-01
16	16	8	2024-12-01
17	17	7	2024-12-01
18	18	6	2024-12-01
19	19	8	2024-12-01
20	20	7	2024-12-01

GUEST

ID	NAME	LASTNAME	PHONENUMBER
1	Tomek	Pietruszka	111222333
2	Adam	Marchewka	123765345
3	Ewa	Trombalska	909845342
4	Alicja	Olszewska	563758965
7	Stuś	Pędziwiatr	001785432
8	Stuś	Pędziwiatr	001785432
9	Katarzyna	Bigos	701928455
11	Andzelika	Lisek	187408223
10	Tomek	Listek	777546012
12	Magdalena	Kot	784085639
13	Adam	Grzybek	097974365
14	Alina	Przybylska	852692095
15	Alina	Przybylska	852692095
16	Oskar	Bartoszyk	890538056

HOUSE KEEPER

employeeid	floorassigned
3	1
7	2
11	3
15	4
19	5

MANAGER

employeeid	Department	BuildingName
1	IT	Building A
5	Human Resources	Building E
9	Maintenance	Building D
13	Food & Beverage	Building C
17	Housekeeping	Building B

PAYMENT

ID	AMENITIESCOST	SERVICECOST	TOTALAMOUNT	PAYMENTDATE	RESERVATIONID
1	500	300	1,750	2024-12-11	1
2	0	250	1,130	2024-12-11	2
3	300	500	1,140	2024-12-11	3
4	0	300	2,300	2024-12-11	4
7	300	400	1,080	2024-12-13	7
6	500	950	2,500	2024-12-11	6
8	300	150	1,450	2024-12-16	8
9	0	0	2,300	2024-12-16	9
10	500	950	6,700	2024-12-16	10
11	300	200	2,675	2024-12-16	11
12	0	300	1,100	2024-12-16	12
13	500	300	1,680	2024-12-16	13
14	0	150	2,310	2024-12-16	14

RECEPTIONIST

employeeid	languages
2	English, Polish
6	German, English
10	Spanish, French
14	Italian, Russian
18	Chinese, Japanese

RESERVATION

ID	GUESTID	ROOMID	CHECKINDATE	CHECKOUTDATE
1	1	19	2024-12-02	2024-12-07
2	2	12	2024-12-04	2024-12-08
3	3	27	2024-12-10	2024-12-14
4	4	26	2024-12-21	2024-12-31
7	9	17	2024-12-27	2024-12-30
6	8	5	2025-01-24	2025-01-31
8	10	29	2024-12-28	2025-01-02
9	11	28	2025-01-06	2025-01-16
10	12	24	2025-01-15	2025-01-30
11	13	20	2025-01-15	2025-01-30
12	14	15	2025-01-26	2025-01-31
13	15	14	2025-02-07	2025-02-15
14	16	13	2025-02-19	2025-02-27

ROOM

ID	ROOMNUMBER	BEDCOUNT	PRICE	ISAVAILABLE	BUILDINGID
1	111	2	160	1	1
2	112	3	210	1	1
3	113	1	95	1	2
4	114	4	240	1	2
5	115	2	150	0	3
6	116	3	200	1	3
7	117	5	290	1	4
8	118	6	350	1	4
11	119	2	175	1	6
12	120	3	220	1	6
13	121	4	270	0	7
14	122	1	110	0	7
15	123	2	160	0	8
16	124	5	320	1	8
17	125	1	95	0	9
18	126	4	250	1	9
19	127	3	190	1	10
20	128	2	145	0	10
21	129	5	305	1	11
22	130	2	160	1	11
23	131	3	190	1	12
24	132	6	350	0	12
25	133	2	140	1	13
26	134	3	200	0	13
27	135	1	85	0	14
28	136	4	230	0	14
29	137	3	200	0	15
30	138	2	170	1	15

SERVICE

ID	SERVICETYPE	TOTALCOST	EMPLOYEEID	RESERVATIONID
1	Cleaning	100	[NULL]	1
2	Food Service	150	[NULL]	1
3	Housekeeping	50	[NULL]	1
4	Cleaning	100	[NULL]	2
5	Food Service	150	[NULL]	2
6	Cleaning	100	[NULL]	3
7	Food Service	150	[NULL]	3
8	Child Care Taking	250	[NULL]	3
9	Cleaning	100	[NULL]	4
10	Food Service	150	[NULL]	4
11	Housekeeping	50	[NULL]	4
18	Food Service	150	[NULL]	7
13	Cleaning	100	[NULL]	6
14	Food Service	150	[NULL]	6
15	Housekeeping	50	[NULL]	6
16	Birthday Surprise	400	[NULL]	6
17	Child Care Taking	250	[NULL]	6
19	Child Care Taking	250	[NULL]	7
20	Food Service	150	[NULL]	8
21	Cleaning	100	[NULL]	10
22	Food Service	150	[NULL]	10
23	Housekeeping	50	[NULL]	10
24	Birthday Surprise	400	[NULL]	10
25	Child Care Taking	250	[NULL]	10
26	Food Service	150	[NULL]	11
27	Housekeeping	50	[NULL]	11
28	Cleaning	100	[NULL]	12
29	Food Service	150	[NULL]	12
30	Housekeeping	50	[NULL]	12
31	Cleaning	100	[NULL]	13
32	Food Service	150	[NULL]	13
33	Housekeeping	50	[NULL]	13
34	Food Service	150	[NULL]	14

USER RATNG

ID	BUILDINGNAME	CITYNAME	RATING	BUILDINGID
1	[NULL]	[NULL]	5	[NULL]
2	Building C	Los Angeles	5	3
3	Building A	Bydgoszcz	3	8
4	Building B	Bydgoszcz	5	9
5	Building D	Los Angeles	2	4
6	[NULL]	Los Angeles	5	[NULL]
7	[NULL]	Bydgoszcz	3	[NULL]
8	Building D	Bydgoszcz	3	11
9	Building A	Brooklyn	4	13
10	Building B	Manhattan	5	6