Brînză Tudor-Alexandru

Grupa 251

Anul 2

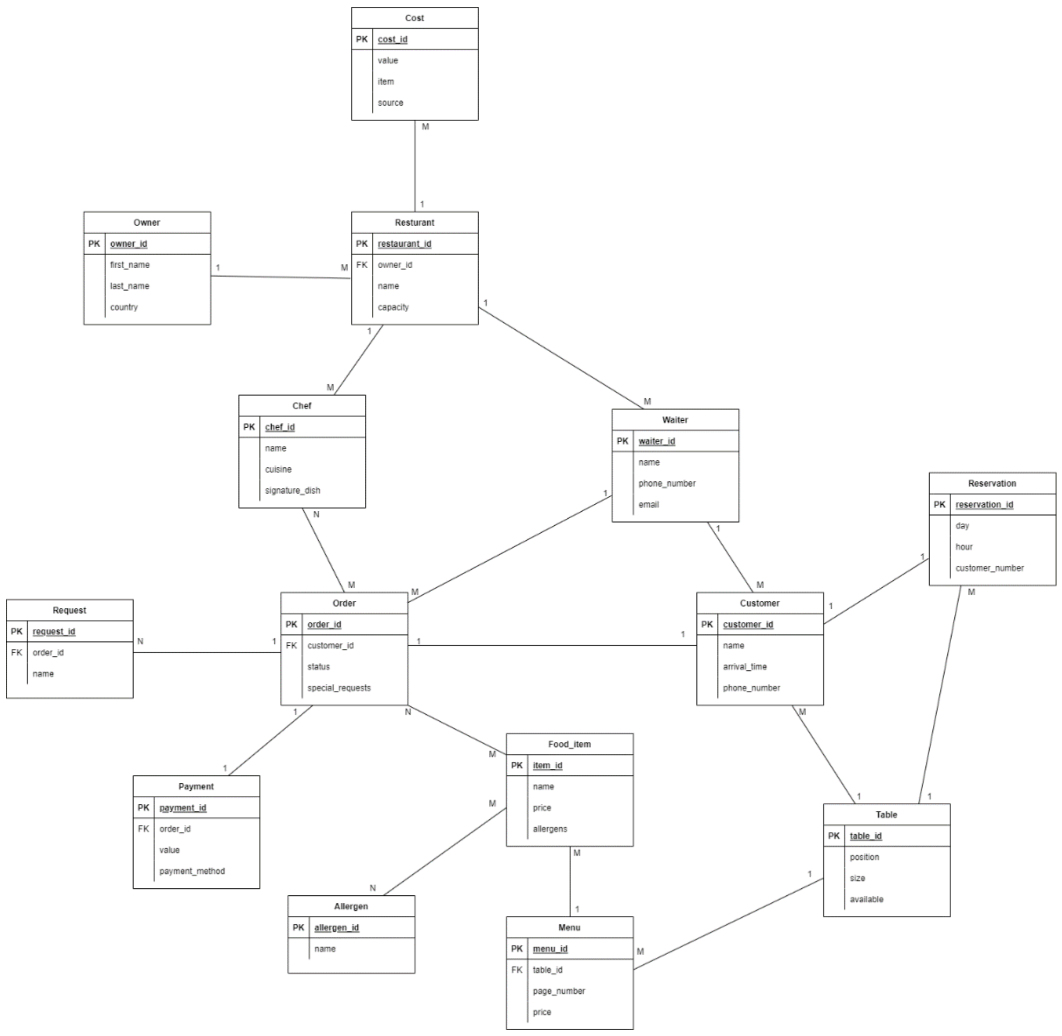
Proiect final SGBD

1.

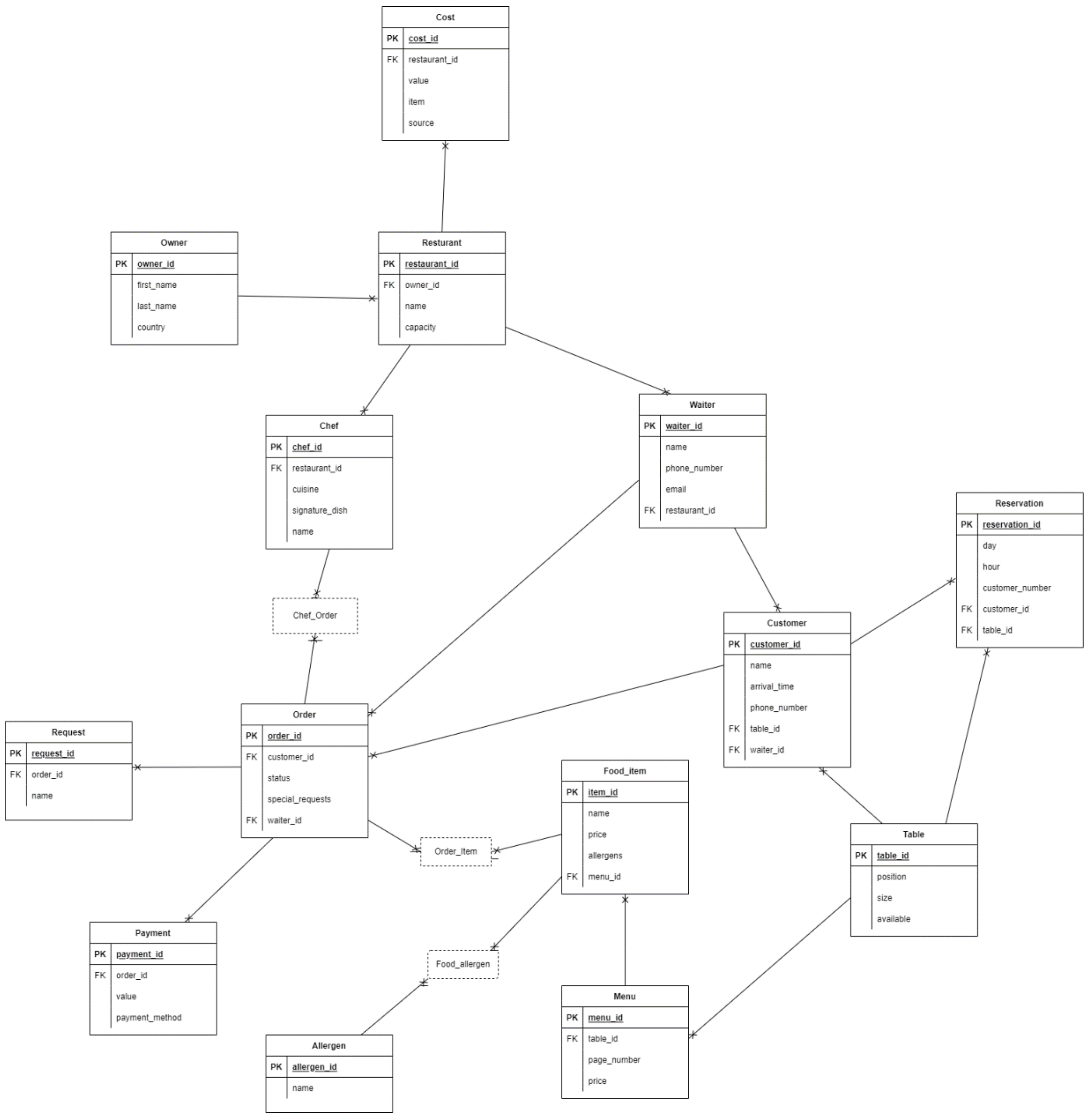
Modelul este relevant deoarece conține modul de funcționare al unui restaurant și poate ajuta la o gestiune mai bună a datelor în cazul în care este personalizat nevoilor specifice restaurantului respectiv.

Fiecare restaurant are un proprietar, dar acest proprietar poate deține mai multe restaurante. Fiecare restaurant are mai multe costuri. Restaurantul are ca angajați bucătari și chelneri. Un restaurant are mai mulți bucătari, iar fiecare bucătar poate lucra la un singur restaurant, același lucru fiind valabil și pentru chelneri. Clienții unui restaurant pot face o rezervare pentru a avea o masă pregătită. Fiecare client va sta la o masă, o masă putând să păstreze mai mulți clienți. Fiecare client va fi servit de un chelner, care mai întâi le va aduce un meniu pentru a se decide asupra comenzii. Chelnerii servesc mai mulți clienți. Din meniu clienții aleg unul sau mai multe produse, apoi le transmit alegerile făcute chelnerului care le adaugă la comandă. Această comandă este apoi transmisă bucătarului care o va pregăti. În final, această comandă genereaza o plată.

2.


Cost
PK cost_id
value
item
source
Owner
PK owner_id
first_name
last_name
country
Resturant
PK restaurant_id
FK owner_id
name
capacity
Chef
PK chef_id
name
cuisine
signature_dish
Waiter
PK waiter_id
name
phone_number
email
Reservation
PK reservation_id
day
hour
customer_number
Request
PK request_id
FK order_id
name
Order
PK order_id
FK customer_id
status
special_requests
Customer
PK customer_id
name
arrival_time
phone_number
Food_item
PK item_id
name
price
allergens
Payment
PK payment_id
FK order_id
value
payment_method
Table
PK table_id
position
size
available
Allergen
PK allergen_id
name
Menu
PK menu_id
FK table_id
page_number
price
M
1
1
M
1
M
1
M
N
M
1
M
1
M
N
1
1
1
1
1
M
1
N
M
M
N
M
1
M
1
1
M
1
M

3.



4.

```
CREATE TABLE OWNER(
    Owner_id INT NOT NULL,
    First_name VARCHAR(20) NOT NULL,
    Last_name VARCHAR(20) NOT NULL,
    Country VARCHAR(20),
    PRIMARY KEY (Owner_id)
);

CREATE TABLE RESTAURANT(
    Restaurant_id INT NOT NULL,
    Owner_id INT NOT NULL,
    Name VARCHAR(20) NOT NULL,
    Capacity INT,
    PRIMARY KEY (Restaurant_id),
    FOREIGN KEY (Owner_id) REFERENCES OWNER(Owner_id)
);

CREATE TABLE COST(
    Cost_id INT NOT NULL,
    Restaurant_id INT NOT NULL,
    Value INT NOT NULL,
    Item VARCHAR(20) NOT NULL,
    Source VARCHAR(20) NOT NULL,
    PRIMARY KEY (Cost_id),
    FOREIGN KEY (Restaurant_id) REFERENCES RESTAURANT(Restaurant_id)
);

CREATE TABLE CHEF(
    Chef_id INT NOT NULL,
    Restaurant_id INT NOT NULL,
    Name VARCHAR(20) NOT NULL,
    Cuisine VARCHAR(20),
    Signature_dish VARCHAR(20),
    PRIMARY KEY (Chef_id),
    FOREIGN KEY (Restaurant_id) REFERENCES RESTAURANT (Restaurant_id)
);

CREATE TABLE WAITER(
    Waiter_id INT NOT NULL,
    Name VARCHAR(20) NOT NULL,
    Phone_number VARCHAR(20),
    Email VARCHAR(20),
    Restaurant_id INT NOT NULL,
    PRIMARY KEY (Waiter_id),
    FOREIGN KEY (Restaurant_id) REFERENCES RESTAURANT(Restaurant_id)
);

CREATE TABLE TABLE_DATA(
    Table_id INT NOT NULL,
    Location VARCHAR(20) NOT NULL,
    Chair_number INT NOT NULL,
    Available INT NOT NULL,
    PRIMARY KEY (Table_id)
);

CREATE TABLE CUSTOMER(
    Customer_id INT NOT NULL,
    Name VARCHAR(20) NOT NULL,
    Arrival_time TIMESTAMP NOT NULL,
    Phone_number VARCHAR(20) NOT NULL,
    Table_id INT NOT NULL,
    Waiter_id INT NOT NULL,
    PRIMARY KEY (Customer_id),
    FOREIGN KEY (Table_id) REFERENCES TABLE_DATA(Table_id),
    FOREIGN KEY (Waiter_id) REFERENCES WAITER(Waiter_id)
);

CREATE TABLE ORDER_DATA(
    Order_id INT NOT NULL,
    Customer_id INT NOT NULL,
    Status INT NOT NULL,
    Waiter_id INT NOT NULL,
    PRIMARY KEY (Order_id),
    FOREIGN KEY (Customer_id) REFERENCES CUSTOMER(Customer_id),
    FOREIGN KEY (Waiter_id) REFERENCES WAITER(Waiter_id)
);

CREATE TABLE REQUEST(
    Request_id INT NOT NULL,
    Name VARCHAR(20) NOT NULL,
    Order_id INT NOT NULL,
    PRIMARY KEY (Request_id),
    FOREIGN KEY (Order_id) REFERENCES ORDER_DATA(Order_id)
```

```
);

CREATE TABLE PAYMENT(
   Payment_id INT NOT NULL,
   Order_id INT NOT NULL,
   Value INT NOT NULL,
   Payment_Method VARCHAR(20),
   PRIMARY KEY (Payment_id),
   FOREIGN KEY (Payment_id) REFERENCES ORDER_DATA (Order_id)
);

CREATE TABLE MENU(
   Menu_id INT NOT NULL,
   Table_id INT NOT NULL,
   Page_number INT,
   Price INT,
   PRIMARY KEY (Menu_id),
   FOREIGN KEY (Table_id) REFERENCES TABLE_DATA (Table_id)
);

CREATE TABLE FOOD_ITEM(
   Item_id INT NOT NULL,
   Name VARCHAR(20) NOT NULL,
   Price INT NOT NULL,
   Menu_id INT NOT NULL,
   PRIMARY KEY (Item_id),
   FOREIGN KEY (Menu_id) REFERENCES MENU (Menu_id)
);

CREATE TABLE RESERVATION(
   Reservation_id INT NOT NULL,
   Arrival_Day DATE NOT NULL,
   Customer_number INT NOT NULL,
   Customer_id INT NOT NULL,
   Table_id INT NOT NULL,
   PRIMARY KEY (Reservation_id),
   FOREIGN KEY (Customer_id) REFERENCES CUSTOMER(Customer_id),
   FOREIGN KEY (Table_id) REFERENCES TABLE_DATA(Table_id)
);

CREATE TABLE ALLERGEN(
   Allergen_id INT NOT NULL,
   Name VARCHAR(20) NOT NULL,
   PRIMARY KEY (Allergen_id)
);

CREATE TABLE CHEF_ORDER(
   Chef_id INT NOT NULL,
   Order_id INT NOT NULL,
   Rating INT,
   PRIMARY KEY(Chef_id, Order_id),
   FOREIGN KEY (Chef_id) REFERENCES CHEF (Chef_id),
   FOREIGN KEY (Order_id) REFERENCES ORDER_DATA (Order_id)
);

CREATE TABLE ORDER_ITEM(
   Order_id INT NOT NULL,
   Item_id INT NOT NULL,
   Quantity INT,
   PRIMARY KEY(Item_id, Order_id),
   FOREIGN KEY (Item_id) REFERENCES FOOD_ITEM (Item_id),
   FOREIGN KEY (Order_id) REFERENCES ORDER_DATA (Order_id)
);

CREATE TABLE FOOD_ALLERGEN(
   Item_id INT NOT NULL,
   Allergen_id INT NOT NULL,
   Percent FLOAT,
   PRIMARY KEY(Item_id, Allergen_id),
   FOREIGN KEY (Item_id) REFERENCES FOOD_ITEM (Item_id),
   FOREIGN KEY (Allergen_id) REFERENCES ALLERGEN (Allergen_id)
);
```

5.

```
CREATE SEQUENCE ID_GENERATOR
START WITH 1
INCREMENT BY 1
```

```
MINVALUE 1

MAXVALUE 100

NOCYCLE;


INSERT INTO OWNER (OWNER_ID, FIRST_NAME, LAST_NAME, COUNTRY)
VALUES (ID_GENERATOR.NEXTVAL,'Ion','Popescu','Romania');

INSERT INTO OWNER (OWNER_ID, FIRST_NAME, LAST_NAME, COUNTRY)
VALUES (ID_GENERATOR.NEXTVAL,'George','Richard','USA');

INSERT INTO OWNER (OWNER_ID, FIRST_NAME, LAST_NAME, COUNTRY)
VALUES (ID_GENERATOR.NEXTVAL,'Pablo','Martinez','Spain');

INSERT INTO OWNER (OWNER_ID, FIRST_NAME, LAST_NAME, COUNTRY)
VALUES (ID_GENERATOR.NEXTVAL,'Adrian','Vasile','Romania');

INSERT INTO OWNER (OWNER_ID, FIRST_NAME, LAST_NAME, COUNTRY)
VALUES (ID_GENERATOR.NEXTVAL,'Joshua','Bradley','England');

INSERT INTO OWNER (OWNER_ID, FIRST_NAME, LAST_NAME, COUNTRY)
VALUES (ID_GENERATOR.NEXTVAL,'Pierre','Aubert','France');

INSERT INTO OWNER (OWNER_ID, FIRST_NAME, LAST_NAME, COUNTRY)
VALUES (ID_GENERATOR.NEXTVAL,'Tobi','Brown','England');


DROP SEQUENCE ID_GENERATOR;


CREATE SEQUENCE ID_GENERATOR

START WITH 1;


INSERT INTO RESTAURANT (RESTAURANT_ID, OWNER_ID, NAME, CAPACITY)
VALUES (ID_GENERATOR.NEXTVAL,4,'Dinner In The Sky',220);

INSERT INTO RESTAURANT (RESTAURANT_ID, OWNER_ID, NAME, CAPACITY)
VALUES (ID_GENERATOR.NEXTVAL,6,'Siena',150);
```

```
INSERT INTO RESTAURANT (RESTAURANT_ID, OWNER_ID, NAME, CAPACITY)
VALUES (ID_GENERATOR.NEXTVAL,1,'The View Lounge',120);

INSERT INTO RESTAURANT (RESTAURANT_ID, OWNER_ID, NAME, CAPACITY)
VALUES (ID_GENERATOR.NEXTVAL,3,'Bankers Hill',90);

INSERT INTO RESTAURANT (RESTAURANT_ID, OWNER_ID, NAME, CAPACITY)
VALUES (ID_GENERATOR.NEXTVAL,2,'El Diablo',110);

INSERT INTO RESTAURANT (RESTAURANT_ID, OWNER_ID, NAME, CAPACITY)
VALUES (ID_GENERATOR.NEXTVAL,4,'Chill Out',200);

INSERT INTO RESTAURANT (RESTAURANT_ID, OWNER_ID, NAME, CAPACITY)
VALUES (ID_GENERATOR.NEXTVAL,5,'Villa Escudero',160);

INSERT INTO RESTAURANT (RESTAURANT_ID, OWNER_ID, NAME, CAPACITY)
VALUES (ID_GENERATOR.NEXTVAL,7,'Princess Heart',50);

INSERT INTO RESTAURANT (RESTAURANT_ID, OWNER_ID, NAME, CAPACITY)
VALUES (ID_GENERATOR.NEXTVAL,5,'Niagara',135);

INSERT INTO RESTAURANT (RESTAURANT_ID, OWNER_ID, NAME, CAPACITY)
VALUES (ID_GENERATOR.NEXTVAL,2,'Ristorante Pizzeria',140s);


DROP SEQUENCE ID_GENERATOR;


CREATE SEQUENCE ID_GENERATOR

START WITH 1;


INSERT INTO COST (COST_ID, RESTAURANT_ID, VALUE, ITEM, SOURCE) VALUES
(ID_GENERATOR.NEXTVAL, 1, 150, 'Salt', 'France');

INSERT INTO COST (COST_ID, RESTAURANT_ID, VALUE, ITEM, SOURCE) VALUES
(ID_GENERATOR.NEXTVAL, 1, 200, 'Beer', 'England');

INSERT INTO COST (COST_ID, RESTAURANT_ID, VALUE, ITEM, SOURCE) VALUES
(ID_GENERATOR.NEXTVAL, 2, 30, 'Sausages', 'Germany');

INSERT INTO COST (COST_ID, RESTAURANT_ID, VALUE, ITEM, SOURCE) VALUES
(ID_GENERATOR.NEXTVAL, 2, 45, 'Cheese', 'Romania');
```

```
INSERT INTO COST (COST_ID, RESTAURANT_ID, VALUE, ITEM, SOURCE) VALUES
(ID_GENERATOR.NEXTVAL, 2, 37, 'Bread', 'France');

INSERT INTO COST (COST_ID, RESTAURANT_ID, VALUE, ITEM, SOURCE) VALUES
(ID_GENERATOR.NEXTVAL, 3, 150, 'Chocolate', 'Belgium');

INSERT INTO COST (COST_ID, RESTAURANT_ID, VALUE, ITEM, SOURCE) VALUES
(ID_GENERATOR.NEXTVAL, 4, 205, 'Tea', 'England');

INSERT INTO COST (COST_ID, RESTAURANT_ID, VALUE, ITEM, SOURCE) VALUES
(ID_GENERATOR.NEXTVAL, 4, 170, 'Truffle', 'Switzerland');

INSERT INTO COST (COST_ID, RESTAURANT_ID, VALUE, ITEM, SOURCE) VALUES
(ID_GENERATOR.NEXTVAL, 5, 125, 'Olive', 'Spain');

INSERT INTO COST (COST_ID, RESTAURANT_ID, VALUE, ITEM, SOURCE) VALUES
(ID_GENERATOR.NEXTVAL, 5, 345, 'Wine', 'Spain');

INSERT INTO COST (COST_ID, RESTAURANT_ID, VALUE, ITEM, SOURCE) VALUES
(ID_GENERATOR.NEXTVAL, 6, 175, 'Sardine', 'Portugal');

INSERT INTO COST (COST_ID, RESTAURANT_ID, VALUE, ITEM, SOURCE) VALUES
(ID_GENERATOR.NEXTVAL, 6, 450, 'Wine', 'France');

INSERT INTO COST (COST_ID, RESTAURANT_ID, VALUE, ITEM, SOURCE) VALUES
(ID_GENERATOR.NEXTVAL, 6, 325, 'Meat', 'Romania');

INSERT INTO COST (COST_ID, RESTAURANT_ID, VALUE, ITEM, SOURCE) VALUES
(ID_GENERATOR.NEXTVAL, 7, 75, 'Potato', 'Thailand');

INSERT INTO COST (COST_ID, RESTAURANT_ID, VALUE, ITEM, SOURCE) VALUES
(ID_GENERATOR.NEXTVAL, 7, 278, 'Vodka', 'Russia');

INSERT INTO COST (COST_ID, RESTAURANT_ID, VALUE, ITEM, SOURCE) VALUES
(ID_GENERATOR.NEXTVAL, 3, 65, 'Biscuits', 'England');

INSERT INTO COST (COST_ID, RESTAURANT_ID, VALUE, ITEM, SOURCE) VALUES
(ID_GENERATOR.NEXTVAL, 4, 105, 'Grapes', 'France');

INSERT INTO COST (COST_ID, RESTAURANT_ID, VALUE, ITEM, SOURCE) VALUES
(ID_GENERATOR.NEXTVAL, 6, 90, 'Tortilla', 'Spain');

INSERT INTO COST (COST_ID, RESTAURANT_ID, VALUE, ITEM, SOURCE) VALUES
(ID_GENERATOR.NEXTVAL, 7, 100, 'Chocolate', 'Belgium');
```

```
DROP SEQUENCE ID_GENERATOR;

CREATE SEQUENCE ID_GENERATOR
START WITH 1;

INSERT INTO CHEF (CHEF_ID, NAME, CUISINE, SIGNATURE_DISH,
RESTAURANT_ID) VALUES (ID_GENERATOR.NEXTVAL, 'George', 'French', 'Coq
Au Vin', 1);
INSERT INTO CHEF (CHEF_ID, NAME, CUISINE, SIGNATURE_DISH,
RESTAURANT_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Frank', 'Brazilian',
'Casquinha De Siri', 1);
INSERT INTO CHEF (CHEF_ID, NAME, CUISINE, SIGNATURE_DISH,
RESTAURANT_ID) VALUES (ID_GENERATOR.NEXTVAL, 'John', 'Caribbean',
'Curried Channa', 2);
INSERT INTO CHEF (CHEF_ID, NAME, CUISINE, SIGNATURE_DISH,
RESTAURANT_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Marius', 'Romanian',
'Pomana Porcului', 3);
INSERT INTO CHEF (CHEF_ID, NAME, CUISINE, SIGNATURE_DISH,
RESTAURANT_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Jack', 'Indian', 'Chicken
Tikka Masala', 4);
INSERT INTO CHEF (CHEF_ID, NAME, CUISINE, SIGNATURE_DISH,
RESTAURANT_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Tobi', 'French', 'Cherry
Clafoutis', 4);
INSERT INTO CHEF (CHEF_ID, NAME, CUISINE, SIGNATURE_DISH,
RESTAURANT_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Simon', 'German',
'Sauerbraten', 4);
INSERT INTO CHEF (CHEF_ID, NAME, CUISINE, SIGNATURE_DISH,
RESTAURANT_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Liam', 'Chinese', 'Fondue
Chinoise', 5);
INSERT INTO CHEF (CHEF_ID, NAME, CUISINE, SIGNATURE_DISH,
RESTAURANT_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Noah', 'Mexican', 'Tacos
Al Pastor', 5);
```

```
INSERT INTO CHEF (CHEF_ID, NAME, CUISINE, SIGNATURE_DISH,
RESTAURANT_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Oliver', 'Italian', 'Osso
Bucco Milanese', 6);

INSERT INTO CHEF (CHEF_ID, NAME, CUISINE, SIGNATURE_DISH,
RESTAURANT_ID) VALUES (ID_GENERATOR.NEXTVAL, 'James', 'Caribbean',
'Fungee And Pepperpot', 6);

INSERT INTO CHEF (CHEF_ID, NAME, CUISINE, SIGNATURE_DISH,
RESTAURANT_ID) VALUES (ID_GENERATOR.NEXTVAL, 'William', 'Indian', 'Gold
Leaf Dosa', 7);

INSERT INTO CHEF (CHEF_ID, NAME, CUISINE, SIGNATURE_DISH,
RESTAURANT_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Lucas', 'Turkish', 'Manti',
7);


DROP SEQUENCE ID_GENERATOR;


CREATE SEQUENCE ID_GENERATOR

START WITH 1;


INSERT INTO WAITER (WAITER_ID, NAME, PHONE_NUMBER, EMAIL,
RESTAURANT_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Maria', '0654254685',
'maria@gmail.com', 1);

INSERT INTO WAITER (WAITER_ID, NAME, PHONE_NUMBER, EMAIL,
RESTAURANT_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Henry', '0628282928',
'henry@gmail.com', 2);

INSERT INTO WAITER (WAITER_ID, NAME, PHONE_NUMBER, EMAIL,
RESTAURANT_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Amelia', '0384927571',
'amelia@gmail.com', 3);

INSERT INTO WAITER (WAITER_ID, NAME, PHONE_NUMBER, EMAIL,
RESTAURANT_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Alexander',
'0168435716', 'alexander@gmail.com', 3);
```

```
INSERT INTO WAITER (WAITER_ID, NAME, PHONE_NUMBER, EMAIL,
RESTAURANT_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Nora', '0935724619',
'nora@gmail.com', 4);

INSERT INTO WAITER (WAITER_ID, NAME, PHONE_NUMBER, EMAIL,
RESTAURANT_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Felix', '07619458329',
'felix@gmail.com', 5);

INSERT INTO WAITER (WAITER_ID, NAME, PHONE_NUMBER, EMAIL,
RESTAURANT_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Levi', '0792872672',
'levi@gmail.com', 6);

INSERT INTO WAITER (WAITER_ID, NAME, PHONE_NUMBER, EMAIL,
RESTAURANT_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Theo', '06696727283',
'theo@gmail.com', 6);

INSERT INTO WAITER (WAITER_ID, NAME, PHONE_NUMBER, EMAIL,
RESTAURANT_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Grace', '028373777279',
'grace@gmail.com', 7);


DROP SEQUENCE ID_GENERATOR;


CREATE SEQUENCE ID_GENERATOR

START WITH 1;


INSERT INTO TABLE_DATA (TABLE_ID, LOCATION, CHAIR_NUMBER, AVAILABLE)
VALUES (ID_GENERATOR.NEXTVAL, 'Smoking Area', 4, 1);

INSERT INTO TABLE_DATA (TABLE_ID, LOCATION, CHAIR_NUMBER, AVAILABLE)
VALUES (ID_GENERATOR.NEXTVAL, 'Open Area', 8, 1);

INSERT INTO TABLE_DATA (TABLE_ID, LOCATION, CHAIR_NUMBER, AVAILABLE)
VALUES (ID_GENERATOR.NEXTVAL, 'Bar Area', 3, 1);

INSERT INTO TABLE_DATA (TABLE_ID, LOCATION, CHAIR_NUMBER, AVAILABLE)
VALUES (ID_GENERATOR.NEXTVAL, 'Smoking Area', 4, 1);

INSERT INTO TABLE_DATA (TABLE_ID, LOCATION, CHAIR_NUMBER, AVAILABLE)
VALUES (ID_GENERATOR.NEXTVAL, 'Open Area', 12, 1);
```

```
INSERT INTO TABLE_DATA (TABLE_ID, LOCATION, CHAIR_NUMBER, AVAILABLE)
VALUES (ID_GENERATOR.NEXTVAL, 'Open Area', 4, 1);

INSERT INTO TABLE_DATA (TABLE_ID, LOCATION, CHAIR_NUMBER, AVAILABLE)
VALUES (ID_GENERATOR.NEXTVAL, 'Smoking Area', 2, 1);

INSERT INTO TABLE_DATA (TABLE_ID, LOCATION, CHAIR_NUMBER, AVAILABLE)
VALUES (ID_GENERATOR.NEXTVAL, 'Bar Area', 4, 1);

INSERT INTO TABLE_DATA (TABLE_ID, LOCATION, CHAIR_NUMBER, AVAILABLE)
VALUES (ID_GENERATOR.NEXTVAL, 'Open Area', 6, 1);

INSERT INTO TABLE_DATA (TABLE_ID, LOCATION, CHAIR_NUMBER, AVAILABLE)
VALUES (ID_GENERATOR.NEXTVAL, 'Smoking Area', 8, 1);

INSERT INTO TABLE_DATA (TABLE_ID, LOCATION, CHAIR_NUMBER, AVAILABLE)
VALUES (ID_GENERATOR.NEXTVAL, 'Open Area', 4, 1);

INSERT INTO TABLE_DATA (TABLE_ID, LOCATION, CHAIR_NUMBER, AVAILABLE)
VALUES (ID_GENERATOR.NEXTVAL, 'Open Area', 8, 1);

INSERT INTO TABLE_DATA (TABLE_ID, LOCATION, CHAIR_NUMBER, AVAILABLE)
VALUES (ID_GENERATOR.NEXTVAL, 'Smoking Area', 4, 1);

INSERT INTO TABLE_DATA (TABLE_ID, LOCATION, CHAIR_NUMBER, AVAILABLE)
VALUES (ID_GENERATOR.NEXTVAL, 'Bar Area', 8, 1);

INSERT INTO TABLE_DATA (TABLE_ID, LOCATION, CHAIR_NUMBER, AVAILABLE)
VALUES (ID_GENERATOR.NEXTVAL, 'Open Area', 10, 1);


DROP SEQUENCE ID_GENERATOR;


CREATE SEQUENCE ID_GENERATOR

START WITH 1;


INSERT INTO CUSTOMER (CUSTOMER_ID, NAME, ARRIVAL_TIME,
PHONE_NUMBER, TABLE_ID, WAITER_ID) VALUES (ID_GENERATOR.NEXTVAL,
'John', TO_TIMESTAMP('2021-11-02 19:30:00', 'YYYY-MM-DD HH24:MI:SS'),
'0248565843', 2, 1);
```

```
INSERT INTO CUSTOMER (CUSTOMER_ID, NAME, ARRIVAL_TIME, PHONE_NUMBER, TABLE_ID, WAITER_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Noah', TO_TIMESTAMP('2021-11-07 19:00:00', 'YYYY-MM-DD HH24:MI:SS'), '0297222757', 1, 1);

INSERT INTO CUSTOMER (CUSTOMER_ID, NAME, ARRIVAL_TIME, PHONE_NUMBER, TABLE_ID, WAITER_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Jennifer', TO_TIMESTAMP('2021-10-20 20:30:00', 'YYYY-MM-DD HH24:MI:SS'), '0378178658', 3, 2);

INSERT INTO CUSTOMER (CUSTOMER_ID, NAME, ARRIVAL_TIME, PHONE_NUMBER, TABLE_ID, WAITER_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Joanna', TO_TIMESTAMP('2021-09-17 21:00:00', 'YYYY-MM-DD HH24:MI:SS'), '0942864716', 4, 2);

INSERT INTO CUSTOMER (CUSTOMER_ID, NAME, ARRIVAL_TIME, PHONE_NUMBER, TABLE_ID, WAITER_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Lisa', TO_TIMESTAMP('2021-08-14 17:30:00', 'YYYY-MM-DD HH24:MI:SS'), '0394678158', 5, 3);

INSERT INTO CUSTOMER (CUSTOMER_ID, NAME, ARRIVAL_TIME, PHONE_NUMBER, TABLE_ID, WAITER_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Bart', TO_TIMESTAMP('2021-09-07 19:30:00', 'YYYY-MM-DD HH24:MI:SS'), '0349768152', 6, 4);

INSERT INTO CUSTOMER (CUSTOMER_ID, NAME, ARRIVAL_TIME, PHONE_NUMBER, TABLE_ID, WAITER_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Mike', TO_TIMESTAMP('2021-09-17 14:00:00', 'YYYY-MM-DD HH24:MI:SS'), '0284869671', 7, 5);

INSERT INTO CUSTOMER (CUSTOMER_ID, NAME, ARRIVAL_TIME, PHONE_NUMBER, TABLE_ID, WAITER_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Connor', TO_TIMESTAMP('2021-07-18 15:30:00', 'YYYY-MM-DD HH24:MI:SS'), '0345687125', 10, 5);

INSERT INTO CUSTOMER (CUSTOMER_ID, NAME, ARRIVAL_TIME, PHONE_NUMBER, TABLE_ID, WAITER_ID) VALUES (ID_GENERATOR.NEXTVAL, 'Asher', TO_TIMESTAMP('2021-08-13 19:00:00', 'YYYY-MM-DD HH24:MI:SS'), '0244589672', 8, 6);

INSERT INTO CUSTOMER (CUSTOMER_ID, NAME, ARRIVAL_TIME, PHONE_NUMBER, TABLE_ID, WAITER_ID) VALUES (ID_GENERATOR.NEXTVAL,
'Bonnie', TO_TIMESTAMP('2021-11-12 22:00:00', 'YYYY-MM-DD HH24:MI:SS'),
'028596748', 9, 6);

INSERT INTO CUSTOMER (CUSTOMER_ID, NAME, ARRIVAL_TIME,
PHONE_NUMBER, TABLE_ID, WAITER_ID) VALUES (ID_GENERATOR.NEXTVAL,
'Rick', TO_TIMESTAMP('2021-12-17 17:00:00', 'YYYY-MM-DD HH24:MI:SS'),
'0285349824', 11, 7);

INSERT INTO CUSTOMER (CUSTOMER_ID, NAME, ARRIVAL_TIME,
PHONE_NUMBER, TABLE_ID, WAITER_ID) VALUES (ID_GENERATOR.NEXTVAL,
'Frank', TO_TIMESTAMP('2021-09-23 16:00:00', 'YYYY-MM-DD HH24:MI:SS'),
'02346984258', 13, 8);

INSERT INTO CUSTOMER (CUSTOMER_ID, NAME, ARRIVAL_TIME,
PHONE_NUMBER, TABLE_ID, WAITER_ID) VALUES (ID_GENERATOR.NEXTVAL,
'Wade', TO_TIMESTAMP('2021-08-30 16:30:00', 'YYYY-MM-DD HH24:MI:SS'),
'02454266974', 12, 8);

INSERT INTO CUSTOMER (CUSTOMER_ID, NAME, ARRIVAL_TIME,
PHONE_NUMBER, TABLE_ID, WAITER_ID) VALUES (ID_GENERATOR.NEXTVAL,
'Trevor', TO_TIMESTAMP('2021-12-14 19:30:00', 'YYYY-MM-DD HH24:MI:SS'),
'013575684265', 15, 9);

INSERT INTO CUSTOMER (CUSTOMER_ID, NAME, ARRIVAL_TIME,
PHONE_NUMBER, TABLE_ID, WAITER_ID) VALUES (ID_GENERATOR.NEXTVAL,
'Milo', TO_TIMESTAMP('2021-10-02 21:00:00', 'YYYY-MM-DD HH24:MI:SS'),
'0942627234', 14, 9);

INSERT INTO CUSTOMER (CUSTOMER_ID, NAME, ARRIVAL_TIME,
PHONE_NUMBER, TABLE_ID, WAITER_ID) VALUES (ID_GENERATOR.NEXTVAL,
'Mike', TO_TIMESTAMP('2021-10-02 20:30:00', 'YYYY-MM-DD HH24:MI:SS'),
'0742423135', 8, 5);


INSERT INTO CUSTOMER (CUSTOMER_ID, NAME, ARRIVAL_TIME,
PHONE_NUMBER, TABLE_ID, WAITER_ID) VALUES (ID_GENERATOR.NEXTVAL,
'Noah', TO_TIMESTAMP('2021-12-05 18:25:00', 'YYYY-MM-DD HH24:MI:SS'),
'0268234717', 8, 4);


DROP SEQUENCE ID_GENERATOR;
```

```
CREATE SEQUENCE ID_GENERATOR

START WITH 1;


INSERT INTO ORDER_DATA (ORDER_ID, CUSTOMER_ID, STATUS, WAITER_ID)
VALUES (ID_GENERATOR.NEXTVAL, 2, 0, 1);

INSERT INTO ORDER_DATA (ORDER_ID, CUSTOMER_ID, STATUS, WAITER_ID)
VALUES (ID_GENERATOR.NEXTVAL, 5, 1, 3);

INSERT INTO ORDER_DATA (ORDER_ID, CUSTOMER_ID, STATUS, WAITER_ID)
VALUES (ID_GENERATOR.NEXTVAL, 8, 0, 5);

INSERT INTO ORDER_DATA (ORDER_ID, CUSTOMER_ID, STATUS, WAITER_ID)
VALUES (ID_GENERATOR.NEXTVAL, 10, 1, 6);

INSERT INTO ORDER_DATA (ORDER_ID, CUSTOMER_ID, STATUS, WAITER_ID)
VALUES (ID_GENERATOR.NEXTVAL, 11, 0, 7);

INSERT INTO ORDER_DATA (ORDER_ID, CUSTOMER_ID, STATUS, WAITER_ID)
VALUES (ID_GENERATOR.NEXTVAL, 15, 1, 9);


DROP SEQUENCE ID_GENERATOR;


CREATE SEQUENCE ID_GENERATOR

START WITH 1;


INSERT INTO PAYMENT (PAYMENT_ID, ORDER_ID, VALUE, PAYMENT_METHOD)
VALUES (ID_GENERATOR.NEXTVAL, 1, 120, 'Cash');

INSERT INTO PAYMENT (PAYMENT_ID, ORDER_ID, VALUE, PAYMENT_METHOD)
VALUES (ID_GENERATOR.NEXTVAL, 3, 250, 'Cash');

INSERT INTO PAYMENT (PAYMENT_ID, ORDER_ID, VALUE, PAYMENT_METHOD)
VALUES (ID_GENERATOR.NEXTVAL, 5, 165, 'Credit Card');
```

```
INSERT INTO PAYMENT (PAYMENT_ID, ORDER_ID, VALUE, PAYMENT_METHOD)
VALUES (ID_GENERATOR.NEXTVAL, 6, 213, 'Cash');

INSERT INTO PAYMENT (PAYMENT_ID, ORDER_ID, VALUE, PAYMENT_METHOD)
VALUES (ID_GENERATOR.NEXTVAL, 2, 56, 'Credit Card');

INSERT INTO PAYMENT (PAYMENT_ID, ORDER_ID, VALUE, PAYMENT_METHOD)
VALUES (ID_GENERATOR.NEXTVAL, 4, 189, 'Credit Card');


DROP SEQUENCE ID_GENERATOR;


CREATE SEQUENCE ID_GENERATOR

START WITH 1;


INSERT INTO RESERVATION (RESERVATION_ID, ARRIVAL_DAY,
CUSTOMER_NUMBER, CUSTOMER_ID, TABLE_ID) VALUES
(ID_GENERATOR.NEXTVAL, TO_DATE('2021-11-02', 'YYYY-MM-DD'), 5, 1, 2);

INSERT INTO RESERVATION (RESERVATION_ID, ARRIVAL_DAY,
CUSTOMER_NUMBER, CUSTOMER_ID, TABLE_ID) VALUES
(ID_GENERATOR.NEXTVAL, TO_DATE('2021-10-20', 'YYYY-MM-DD'), 2, 3, 3);

INSERT INTO RESERVATION (RESERVATION_ID, ARRIVAL_DAY,
CUSTOMER_NUMBER, CUSTOMER_ID, TABLE_ID) VALUES
(ID_GENERATOR.NEXTVAL, TO_DATE('2021-08-14', 'YYYY-MM-DD'), 3, 5, 5);

INSERT INTO RESERVATION (RESERVATION_ID, ARRIVAL_DAY,
CUSTOMER_NUMBER, CUSTOMER_ID, TABLE_ID) VALUES
(ID_GENERATOR.NEXTVAL, TO_DATE('2021-07-18', 'YYYY-MM-DD'), 4, 8, 10);

INSERT INTO RESERVATION (RESERVATION_ID, ARRIVAL_DAY,
CUSTOMER_NUMBER, CUSTOMER_ID, TABLE_ID) VALUES
(ID_GENERATOR.NEXTVAL, TO_DATE('2021-11-12', 'YYYY-MM-DD'), 1, 10, 9);

INSERT INTO RESERVATION (RESERVATION_ID, ARRIVAL_DAY,
CUSTOMER_NUMBER, CUSTOMER_ID, TABLE_ID) VALUES
(ID_GENERATOR.NEXTVAL, TO_DATE('2021-09-23', 'YYYY-MM-DD'), 3, 12, 13);

DROP SEQUENCE ID_GENERATOR;
```

```
CREATE SEQUENCE ID_GENERATOR

START WITH 1;


INSERT INTO MENU (MENU_ID, TABLE_ID, PAGE_NUMBER, PRICE) VALUES
(ID_GENERATOR.NEXTVAL, 2, 50, 15);

INSERT INTO MENU (MENU_ID, TABLE_ID, PAGE_NUMBER, PRICE) VALUES
(ID_GENERATOR.NEXTVAL, 5, 60, 20);

INSERT INTO MENU (MENU_ID, TABLE_ID, PAGE_NUMBER, PRICE) VALUES
(ID_GENERATOR.NEXTVAL, 7, 25, 10);

INSERT INTO MENU (MENU_ID, TABLE_ID, PAGE_NUMBER, PRICE) VALUES
(ID_GENERATOR.NEXTVAL, 8, 30, 12);

INSERT INTO MENU (MENU_ID, TABLE_ID, PAGE_NUMBER, PRICE) VALUES
(ID_GENERATOR.NEXTVAL, 10, 50, 15);

INSERT INTO MENU (MENU_ID, TABLE_ID, PAGE_NUMBER, PRICE) VALUES
(ID_GENERATOR.NEXTVAL, 12, 25, 10);

INSERT INTO MENU (MENU_ID, TABLE_ID, PAGE_NUMBER, PRICE) VALUES
(ID_GENERATOR.NEXTVAL, 15, 30, 12);


DROP SEQUENCE ID_GENERATOR;


CREATE SEQUENCE ID_GENERATOR

START WITH 1;


INSERT INTO FOOD_ITEM (ITEM_ID, NAME, PRICE, MENU_ID) VALUES
(ID_GENERATOR.NEXTVAL, 'Steak', 50, 1);

INSERT INTO FOOD_ITEM (ITEM_ID, NAME, PRICE, MENU_ID) VALUES
(ID_GENERATOR.NEXTVAL, 'Chicken Wings', 30, 1);
```

```
INSERT INTO FOOD_ITEM (ITEM_ID, NAME, PRICE, MENU_ID) VALUES
(ID_GENERATOR.NEXTVAL, 'Fries', 10, 2);

INSERT INTO FOOD_ITEM (ITEM_ID, NAME, PRICE, MENU_ID) VALUES
(ID_GENERATOR.NEXTVAL, 'Garlic Sauce', 5, 3);

INSERT INTO FOOD_ITEM (ITEM_ID, NAME, PRICE, MENU_ID) VALUES
(ID_GENERATOR.NEXTVAL, 'Pizza', 25, 3);

INSERT INTO FOOD_ITEM (ITEM_ID, NAME, PRICE, MENU_ID) VALUES
(ID_GENERATOR.NEXTVAL, 'Soup', 15, 4);

INSERT INTO FOOD_ITEM (ITEM_ID, NAME, PRICE, MENU_ID) VALUES
(ID_GENERATOR.NEXTVAL, 'Noodles', 25, 5);

INSERT INTO FOOD_ITEM (ITEM_ID, NAME, PRICE, MENU_ID) VALUES
(ID_GENERATOR.NEXTVAL, 'Nuggets', 20, 6);

INSERT INTO FOOD_ITEM (ITEM_ID, NAME, PRICE, MENU_ID) VALUES
(ID_GENERATOR.NEXTVAL, 'Quesadilla', 50, 6);

INSERT INTO FOOD_ITEM (ITEM_ID, NAME, PRICE, MENU_ID) VALUES
(ID_GENERATOR.NEXTVAL, 'Rice', 10, 7);


INSERT INTO CHEF_ORDER (CHEF_ID, ORDER_ID, RATING) VALUES (1, 2, 7);

INSERT INTO CHEF_ORDER (CHEF_ID, ORDER_ID, RATING) VALUES (3, 1, 8);

INSERT INTO CHEF_ORDER (CHEF_ID, ORDER_ID, RATING) VALUES (5, 5, 5);

INSERT INTO CHEF_ORDER (CHEF_ID, ORDER_ID, RATING) VALUES (8, 4, 10);

INSERT INTO CHEF_ORDER (CHEF_ID, ORDER_ID, RATING) VALUES (10, 6, 9);

INSERT INTO CHEF_ORDER (CHEF_ID, ORDER_ID, RATING) VALUES (12, 3, 6);


INSERT INTO ORDER_ITEM (ORDER_ID, ITEM_ID, QUANTITY) VALUES (1, 1, 2);

INSERT INTO ORDER_ITEM (ORDER_ID, ITEM_ID, QUANTITY) VALUES (1, 3, 1);

INSERT INTO ORDER_ITEM (ORDER_ID, ITEM_ID, QUANTITY) VALUES (1, 10, 1);

INSERT INTO ORDER_ITEM (ORDER_ID, ITEM_ID, QUANTITY) VALUES (2, 5, 1);

INSERT INTO ORDER_ITEM (ORDER_ID, ITEM_ID, QUANTITY) VALUES (3, 2, 1);
INSERT INTO ORDER_ITEM (ORDER_ID, ITEM_ID, QUANTITY) VALUES (4, 6, 1);

INSERT INTO ORDER_ITEM (ORDER_ID, ITEM_ID, QUANTITY) VALUES (5, 3, 1);

INSERT INTO ORDER_ITEM (ORDER_ID, ITEM_ID, QUANTITY) VALUES (5, 4, 1);

INSERT INTO ORDER_ITEM (ORDER_ID, ITEM_ID, QUANTITY) VALUES (6, 5, 2);

INSERT INTO ORDER_ITEM (ORDER_ID, ITEM_ID, QUANTITY) VALUES (6, 6, 1);


DROP SEQUENCE ID_GENERATOR;


CREATE SEQUENCE ID_GENERATOR

START WITH 1;


INSERT INTO REQUEST (REQUEST_ID, NAME, ORDER_ID) VALUES
(ID_GENERATOR.NEXTVAL, 'Medium Rare', 1);

INSERT INTO REQUEST (REQUEST_ID, NAME, ORDER_ID) VALUES
(ID_GENERATOR.NEXTVAL, 'No Crust', 3);

INSERT INTO REQUEST (REQUEST_ID, NAME, ORDER_ID) VALUES
(ID_GENERATOR.NEXTVAL, 'No Olives', 2);

INSERT INTO REQUEST (REQUEST_ID, NAME, ORDER_ID) VALUES
(ID_GENERATOR.NEXTVAL, 'No Onion', 2);

INSERT INTO REQUEST (REQUEST_ID, NAME, ORDER_ID) VALUES
(ID_GENERATOR.NEXTVAL, 'No Bread', 4);

INSERT INTO REQUEST (REQUEST_ID, NAME, ORDER_ID) VALUES
(ID_GENERATOR.NEXTVAL, 'Extra Garlic', 5);

INSERT INTO REQUEST (REQUEST_ID, NAME, ORDER_ID) VALUES
(ID_GENERATOR.NEXTVAL, 'No Mushrooms', 6);

INSERT INTO REQUEST (REQUEST_ID, NAME, ORDER_ID) VALUES
(ID_GENERATOR.NEXTVAL, 'Extra Corn', 6);


DROP SEQUENCE ID_GENERATOR;
```

```
CREATE SEQUENCE ID_GENERATOR

START WITH 1;


INSERT INTO ALLERGEN (ALLERGEN_ID, NAME) VALUES
(ID_GENERATOR.NEXTVAL, 'Celery');

INSERT INTO ALLERGEN (ALLERGEN_ID, NAME) VALUES
(ID_GENERATOR.NEXTVAL, 'Gluten');

INSERT INTO ALLERGEN (ALLERGEN_ID, NAME) VALUES
(ID_GENERATOR.NEXTVAL, 'Egg');

INSERT INTO ALLERGEN (ALLERGEN_ID, NAME) VALUES
(ID_GENERATOR.NEXTVAL, 'Fish');

INSERT INTO ALLERGEN (ALLERGEN_ID, NAME) VALUES
(ID_GENERATOR.NEXTVAL, 'Peanuts');

INSERT INTO ALLERGEN (ALLERGEN_ID, NAME) VALUES
(ID_GENERATOR.NEXTVAL, 'Soybeans');


INSERT INTO FOOD_ALLERGEN (ITEM_ID, ALLERGEN_ID, PERCENT) VALUES (2,
5, 0.20);

INSERT INTO FOOD_ALLERGEN (ITEM_ID, ALLERGEN_ID, PERCENT) VALUES (2,
6, 0.35);

INSERT INTO FOOD_ALLERGEN (ITEM_ID, ALLERGEN_ID, PERCENT) VALUES (2,
2, 1.00);

INSERT INTO FOOD_ALLERGEN (ITEM_ID, ALLERGEN_ID, PERCENT) VALUES (5,
2, 10.00);

INSERT INTO FOOD_ALLERGEN (ITEM_ID, ALLERGEN_ID, PERCENT) VALUES (6,
1, 0.35);

INSERT INTO FOOD_ALLERGEN (ITEM_ID, ALLERGEN_ID, PERCENT) VALUES (7,
2, 2.35);
```

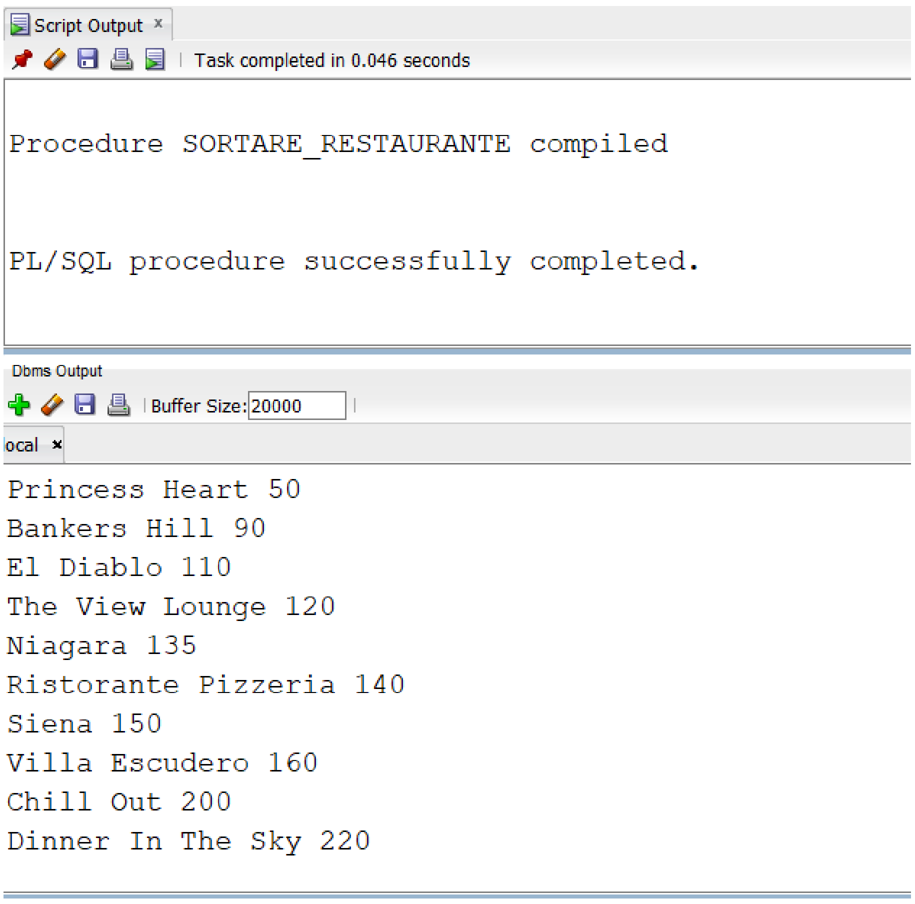
```
INSERT INTO FOOD_ALLERGEN (ITEM_ID, ALLERGEN_ID, PERCENT) VALUES (4,
6, 0.12);
INSERT INTO FOOD_ALLERGEN (ITEM_ID, ALLERGEN_ID, PERCENT) VALUES (4,
2, 0.17);
INSERT INTO FOOD_ALLERGEN (ITEM_ID, ALLERGEN_ID, PERCENT) VALUES (9,
2, 1.15);
INSERT INTO FOOD_ALLERGEN (ITEM_ID, ALLERGEN_ID, PERCENT) VALUES (9,
1, 0.25);
INSERT INTO FOOD_ALLERGEN (ITEM_ID, ALLERGEN_ID, PERCENT) VALUES (9,
6, 0.09);
```

6. Sortati restaurantele crescator dupa capacitate.

```
CREATE OR REPLACE TYPE vector IS VARRAY(20) OF VARCHAR(20);
/
CREATE OR REPLACE TYPE tablou_imbricat IS TABLE OF NUMBER;
/
CREATE OR REPLACE PROCEDURE sortare_restaurante
IS
   nume_restaurant vector := vector();
   capacitate tablou_imbricat := tablou_imbricat();
   gasit NUMBER;
   aux_number NUMBER;
   aux_char VARCHAR(20);
BEGIN

   SELECT name
   BULK COLLECT INTO nume_restaurant
```

```
FROM restaurant;

SELECT capacity
BULK COLLECT INTO capacitate
FROM restaurant;

gasit := 0;

WHILE gasit = 0 LOOP

   gasit := 1;

   FOR j IN capacitate.FIRST..capacitate.LAST - 1 LOOP
      IF capacitate(j) > capacitate(j + 1) THEN
         aux_number := capacitate(j);
         capacitate(j) := capacitate(j + 1);
         capacitate(j + 1) := aux_number;

         aux_char := nume_restaurant(j);
         nume_restaurant(j) := nume_restaurant(j + 1);
         nume_restaurant(j + 1) := aux_char;

         gasit := 0;
      END IF;
   END LOOP;
```

```
    END LOOP;

    FOR j IN nume_restaurant.FIRST..nume_restaurant.LAST LOOP
        DBMS_OUTPUT.PUT_LINE (nume_restaurant(j) || ' ' || capacitate(j));
    END LOOP;

END sortare_restaurante;

/
BEGIN
    sortare_restaurante;
END;
```

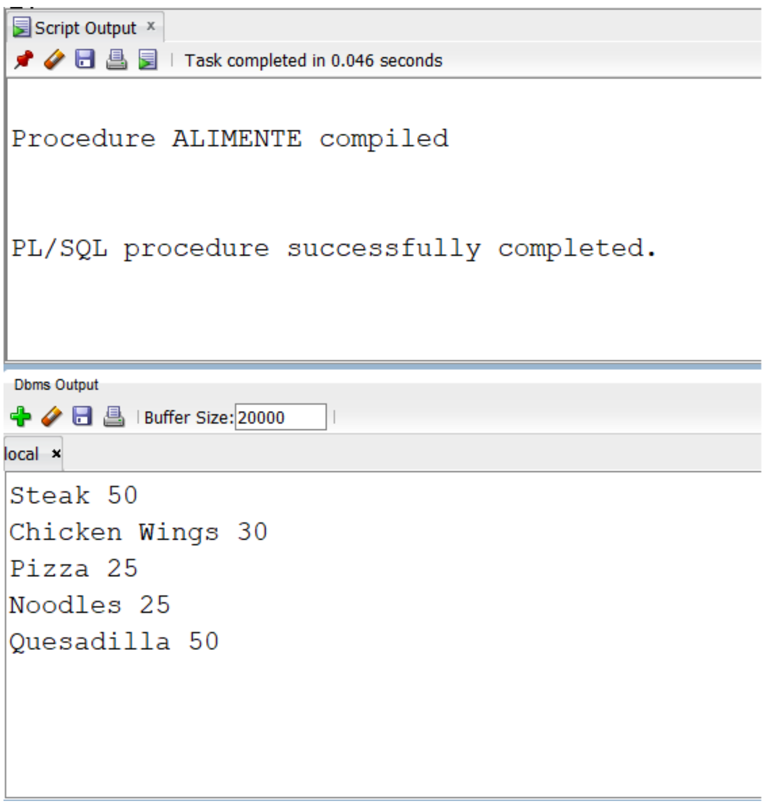


```
Procedure SORTARE_RESTAURANTE compiled


PL/SQL procedure successfully completed.
```

```
Princess Heart 50
Bankers Hill 90
El Diablo 110
The View Lounge 120
Niagara 135
Ristorante Pizzeria 140
Siena 150
Villa Escudero 160
Chill Out 200
Dinner In The Sky 220
```

7. Afisati doar mancarurile care costa cel putin 25 de lei. – ciclu cursor

```
CREATE OR REPLACE PROCEDURE alimente
IS
   CURSOR c IS
      SELECT *
      FROM FOOD_ITEM
      WHERE price >= 25;
BEGIN

   FOR aliment IN c LOOP
      DBMS_OUTPUT.PUT_LINE(aliment.name || ' ' || aliment.price);
   END LOOP;

END alimente;

/
BEGIN
   alimente;
END;
```



```
Procedure ALIMENTE compiled


PL/SQL procedure successfully completed.
```

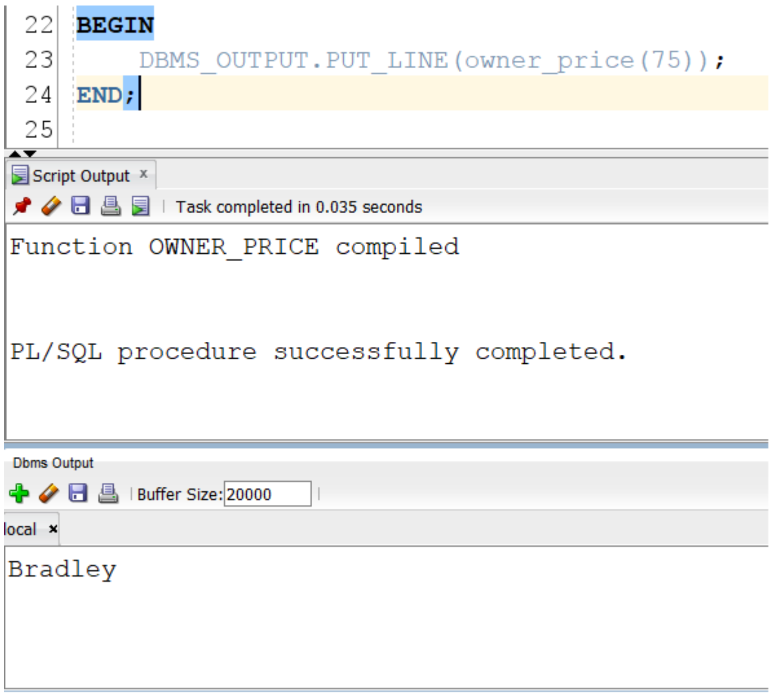
```
Steak 50
Chicken Wings 30
Pizza 25
Noodles 25
Quesadilla 50
```

8. Afisati numele de familie al proprietarului restaurantului care cumpara un produs cu un pret dat ca parametru. Daca exista mai multe produse cu acel pret sau daca nu exista produs cu acel pret se va afisa o eroare.

```
CREATE OR REPLACE FUNCTION owner_price (price COST.VALUE%TYPE)
RETURN VARCHAR IS
   nume OWNER.LAST_NAME%TYPE;
BEGIN
   SELECT o.LAST_NAME INTO nume
   FROM RESTAURANT r JOIN OWNER o ON (o.OWNER_ID = r.OWNER_ID)
                 JOIN COST c ON (c.RESTAURANT_ID = r.RESTAURANT_ID)
   WHERE c.VALUE = price;

   RETURN nume;
   EXCEPTION
      WHEN NO_DATA_FOUND THEN
         RAISE_APPLICATION_ERROR(-20000, 'Nu exista niciun produs cu pretul
indicat!');
      WHEN TOO_MANY_ROWS THEN
         RAISE_APPLICATION_ERROR(-20001, 'Exista mai multe produse cu pretul
indicat!');
      WHEN OTHERS THEN
         RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');

END owner_price;
```

Apelam functia cu parametrul 75, valoare care exista in tabel o singura data.

```
BEGIN
   DBMS_OUTPUT.PUT_LINE(owner_price(375));
END;
```



```
BEGIN
    DBMS_OUTPUT.PUT_LINE(owner_price(75));
END;
```

Script Output ×

Task completed in 0.035 seconds

```
Function OWNER_PRICE compiled


PL/SQL procedure successfully completed.
```

Dbms Output

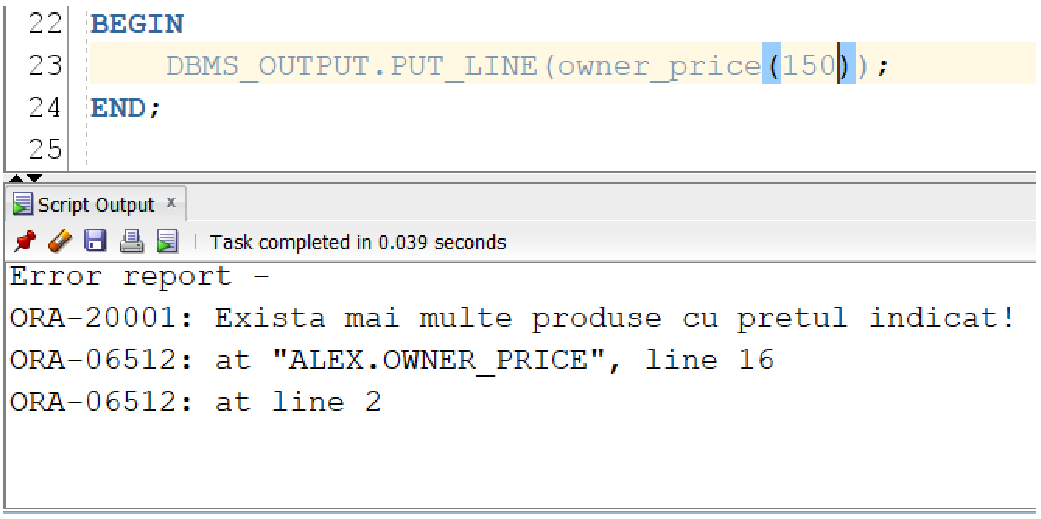
Buffer Size: 20000

local ×

```
Bradley
```

Apelam functia cu parametrul 150, valoare care apare de 2 ori in tabel.

```
BEGIN
   DBMS_OUTPUT.PUT_LINE(owner_price(150));
END;
```



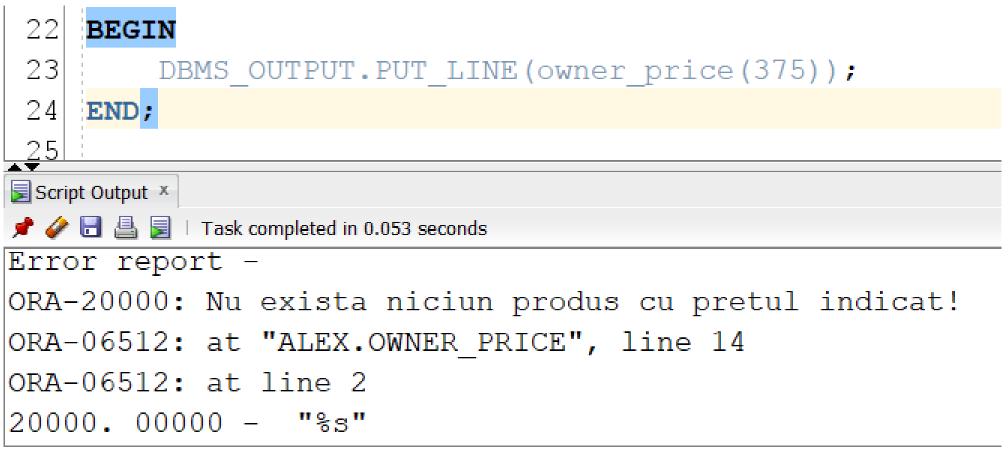
```
BEGIN
    DBMS_OUTPUT.PUT_LINE(owner_price(150));
END;
```

Script Output

Task completed in 0.039 seconds

```
Error report -
ORA-20001: Exista mai multe produse cu pretul indicat!
ORA-06512: at "ALEX.OWNER_PRICE", line 16
ORA-06512: at line 2
```

Apelam functia cu parametrul 375, valoare care nu apare in tabel.



```
BEGIN
    DBMS_OUTPUT.PUT_LINE(owner_price(375));
END;
```

Script Output

Task completed in 0.053 seconds

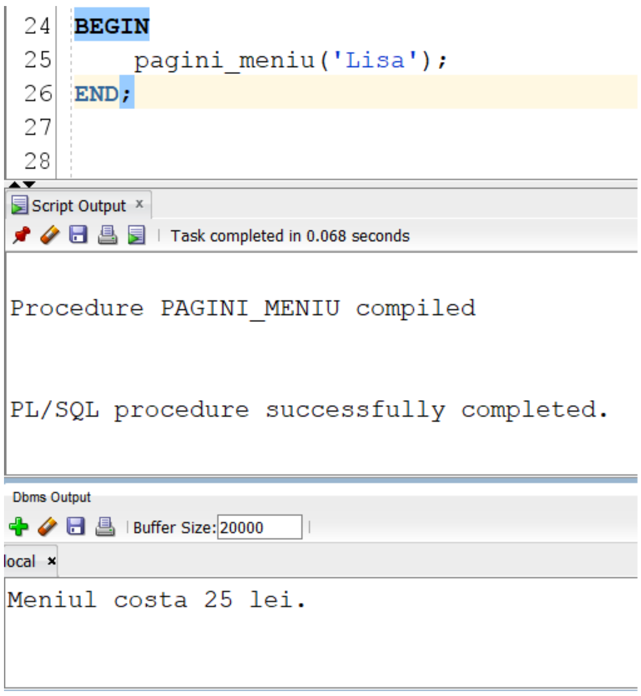
```
Error report -
ORA-20000: Nu exista niciun produs cu pretul indicat!
ORA-06512: at "ALEX.OWNER_PRICE", line 14
ORA-06512: at line 2
20000. 00000 -  "%s"
```

9. Se da ca parametru numele unui client si se afiseaza numarul de pagini ale meniului pe care acesta l-a primit. Daca exista mai multi clienti sau daca nu exista un client cu acel nume se va afisa o eroare.

```
CREATE OR REPLACE PROCEDURE pagini_meniu (nume
CUSTOMER.NAME%TYPE)
IS numar_pagini MENU.MENU_ID%TYPE;

BEGIN
   SELECT m.PAGE_NUMBER INTO numar_pagini
   FROM CUSTOMER c JOIN ORDER_DATA od ON (c.CUSTOMER_ID =
od.CUSTOMER_ID)
               JOIN ORDER_ITEM oi ON (od.ORDER_ID = oi.ORDER_ID)
               JOIN FOOD_ITEM f   ON (f.ITEM_ID = oi.ITEM_ID)
               JOIN MENU m        ON (m.MENU_ID  = f.MENU_ID)
   WHERE c.NAME = nume;

   DBMS_OUTPUT.PUT_LINE('Meniul costa ' || numar_pagini || ' lei.');

   EXCEPTION
      WHEN NO_DATA_FOUND THEN
         RAISE_APPLICATION_ERROR(-20000, 'Nu exista client cu acest nume sau
clientul nu a efectuat o comanda!');
      WHEN TOO_MANY_ROWS THEN
         RAISE_APPLICATION_ERROR(-20001, 'Exista mai multi clienti cu acest
nume!');
      WHEN OTHERS THEN
         RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');

END pagini_meniu;
```

Apelam procedura cu numele Lisa, nume care apare o singura data in tabel.

```
BEGIN
   pagini_meniu('Lisa');
END;
```



```
BEGIN
    pagini_meniu('Lisa');
END;
```

Script Output ×

Task completed in 0.068 seconds

```
Procedure PAGINI_MENIU compiled


PL/SQL procedure successfully completed.
```

Dbms Output

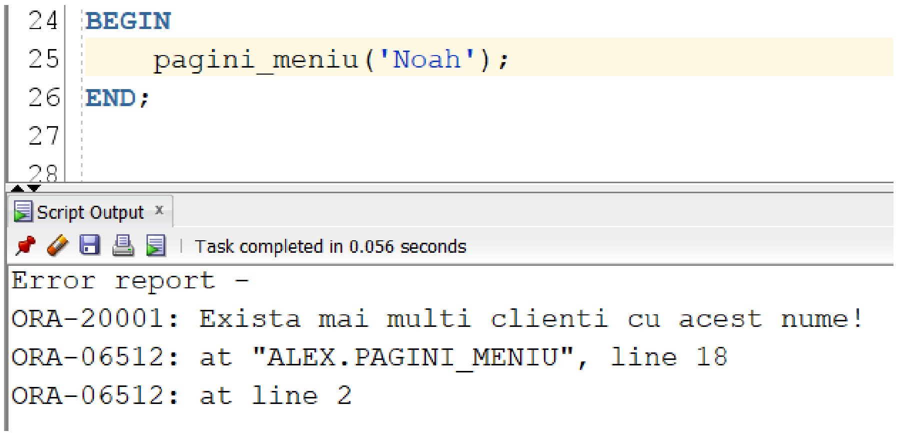
Buffer Size: 20000

local ×

```
Meniul costa 25 lei.
```

Apelam functia cu parametrul Noah, nume care apare de 2 ori in tabel.

```
BEGIN
   pagini_meniu('Noah');
END;
```



```
BEGIN
    pagini_meniu('Noah');
END;
```

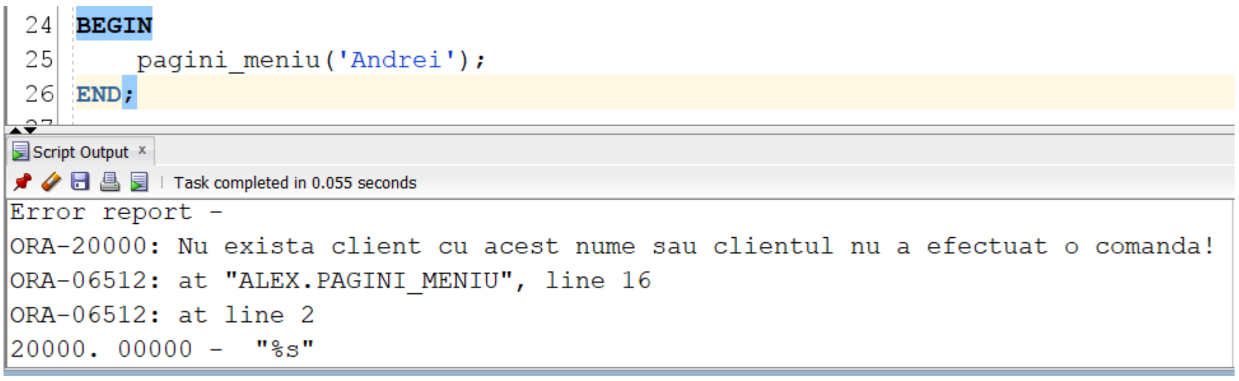
Script Output ×

Task completed in 0.056 seconds

```
Error report -
ORA-20001: Exista mai multi clienti cu acest nume!
ORA-06512: at "ALEX.PAGINI_MENIU", line 18
ORA-06512: at line 2
```

Apelam functia cu parametrul Andrei, nume care nu exista in tabel.

```
BEGIN
   pagini_meniu('Andrei');
END;
```



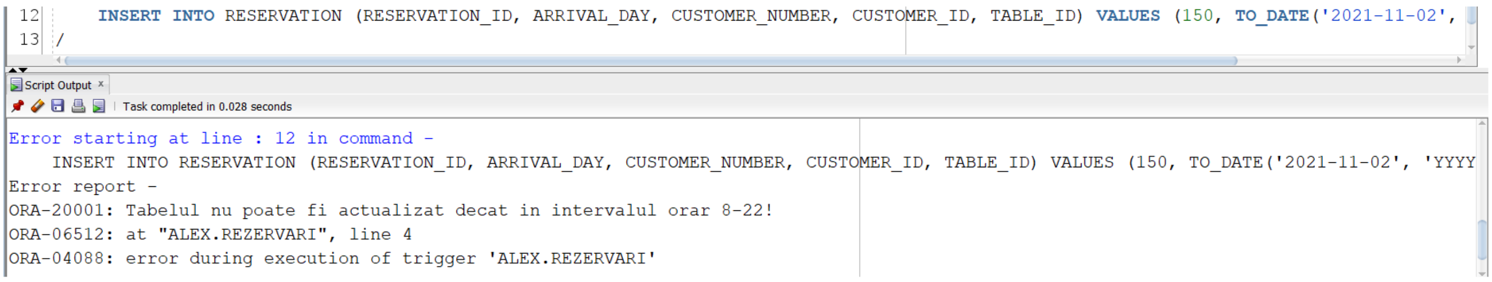
```
BEGIN
    pagini_meniu('Andrei');
END;
```

```
Error report -
ORA-20000: Nu exista client cu acest nume sau clientul nu a efectuat o comanda!
ORA-06512: at "ALEX.PAGINI_MENIU", line 16
ORA-06512: at line 2
20000. 00000 -  "%s"
```

10.Creati un trigger care sa nu permita modificarea rezervarilor decat in timpul programului de functionare al restaurantelor(8:00 – 22:00) pentru a asigura verificarea disponibilitatii.

```
CREATE OR REPLACE TRIGGER rezervari
BEFORE INSERT OR UPDATE OR DELETE ON RESERVATION

BEGIN

   IF TO_CHAR(SYSDATE,'HH24') NOT BETWEEN 8 AND 22 THEN
      RAISE_APPLICATION_ERROR(-20001,'Tabelul nu poate fi actualizat decat in intervalul orar 8-22!');
   END IF;

END rezervari;
```

Incercam sa inseram in tabelul Reservation in afara intervalului orar.



```
    INSERT INTO RESERVATION (RESERVATION_ID, ARRIVAL_DAY, CUSTOMER_NUMBER, CUSTOMER_ID, TABLE_ID) VALUES (150, TO_DATE('2021-11-02',
/
```

Script Output ×

Task completed in 0.028 seconds

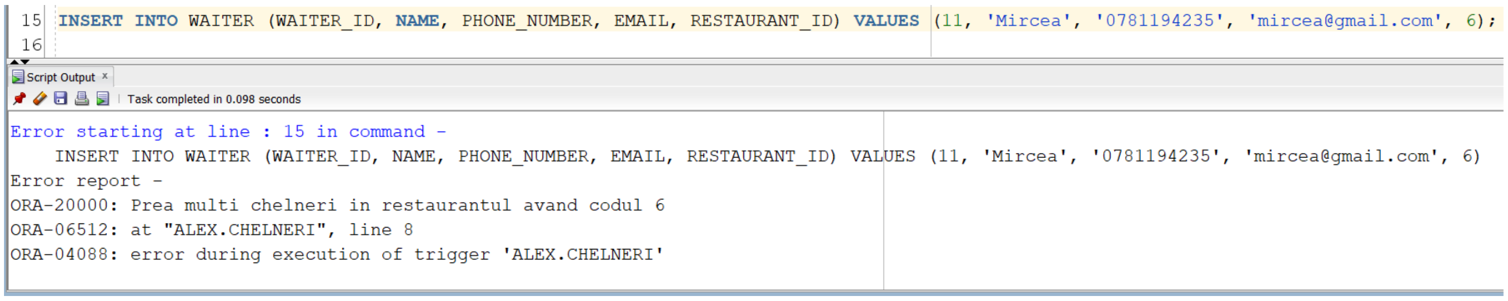
```
Error starting at line : 12 in command -
    INSERT INTO RESERVATION (RESERVATION_ID, ARRIVAL_DAY, CUSTOMER_NUMBER, CUSTOMER_ID, TABLE_ID) VALUES (150, TO_DATE('2021-11-02', 'YYYY
Error report -
ORA-20001: Tabelul nu poate fi actualizat decat in intervalul orar 8-22!
ORA-06512: at "ALEX.REZERVARI", line 4
ORA-04088: error during execution of trigger 'ALEX.REZERVARI'
```

11. Creati un trigger care sa nu permita mai mult de 3 chelneri angajati la un anumit restaurant.

```
CREATE OR REPLACE TRIGGER chelneri
   BEFORE INSERT OR UPDATE OF RESTAURANT_ID ON WAITER
   FOR EACH ROW
DECLARE
   numar_chelneri NUMBER;
BEGIN
   SELECT COUNT(*) INTO numar_chelneri
   FROM WAITER
   WHERE RESTAURANT_ID = :NEW.RESTAURANT_ID;
   IF numar_chelneri = 3 THEN
      RAISE_APPLICATION_ERROR(-20000,'Prea multi chelneri in restaurantul avand codul ' || :NEW.RESTAURANT_ID);
   END IF;
END chelneri;
```

Incercam sa inseram un nou chelner la restaurantul cu id-ul 6, unde exista deja 3 chelneri angajati.

```
INSERT INTO WAITER (WAITER_ID, NAME, PHONE_NUMBER, EMAIL,
RESTAURANT_ID) VALUES (11, 'Mircea', '0781194235', 'mircea@gmail.com', 6);
```



```
INSERT INTO WAITER (WAITER_ID, NAME, PHONE_NUMBER, EMAIL, RESTAURANT_ID) VALUES (11, 'Mircea', '0781194235', 'mircea@gmail.com', 6);
```

```
Error starting at line : 15 in command -
    INSERT INTO WAITER (WAITER_ID, NAME, PHONE_NUMBER, EMAIL, RESTAURANT_ID) VALUES (11, 'Mircea', '0781194235', 'mircea@gmail.com', 6)
Error report -
ORA-20000: Prea multi chelneri in restaurantul avand codul 6
ORA-06512: at "ALEX.CHELNERI", line 8
ORA-04088: error during execution of trigger 'ALEX.CHELNERI'
```

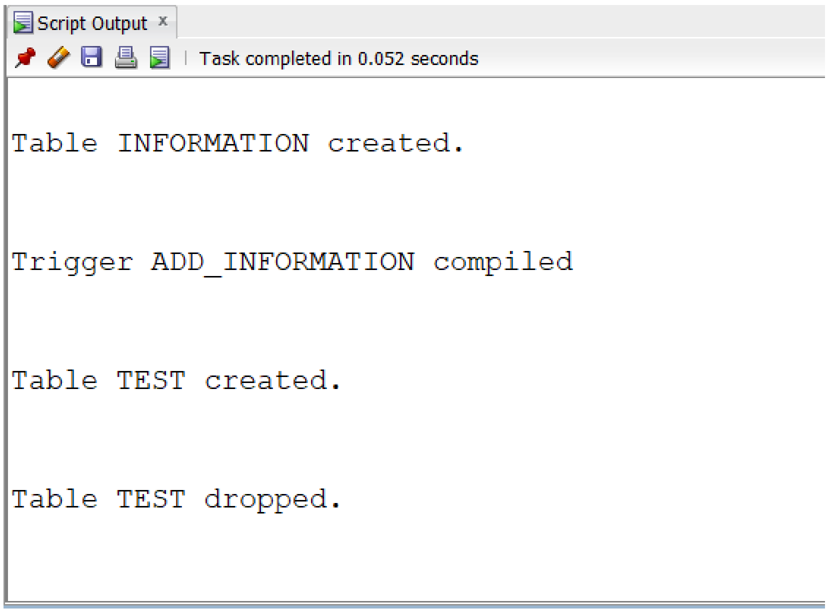
12. Creati un tabel care sa monitorizeze adaugarea de informatii in baza de date. Folositi un trigger pentru a popula acest tabel.

```
CREATE TABLE INFORMATION(
   username VARCHAR2(30),
   event VARCHAR2(20),
   object_name VARCHAR2(30),
   date_of_event DATE
);
/
CREATE OR REPLACE TRIGGER add_information
AFTER CREATE OR ALTER OR DROP ON SCHEMA

BEGIN

   INSERT INTO INFORMATION
   VALUES (SYS.LOGIN_USER, SYS.SYSEVENT, SYS.DICTIONARY_OBJ_NAME,
SYSDATE);
```

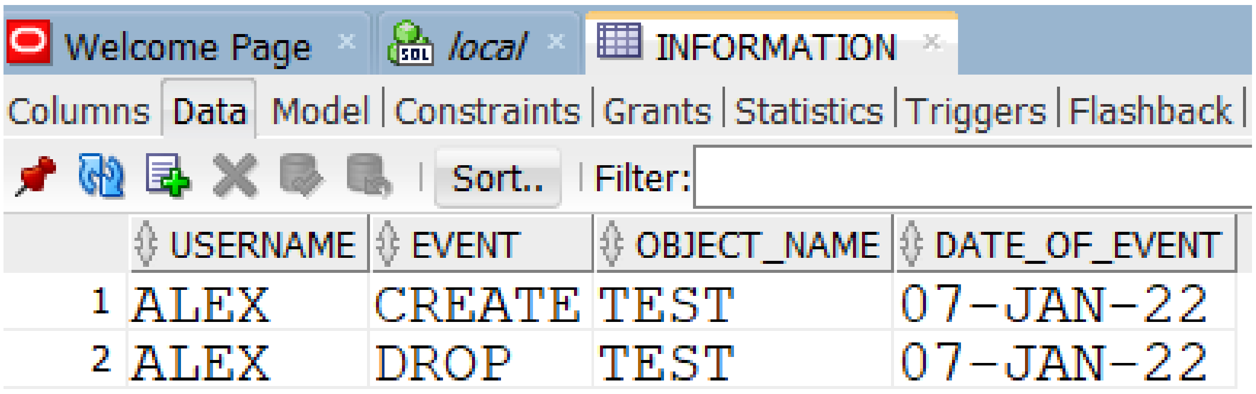
```
END add_information;
```

Pentru a testa cream un tabel si apoi il eliminam.

```
CREATE TABLE TEST(
   random_row NUMBER
);
/
DROP TABLE TEST;
```



```
Table INFORMATION created.


Trigger ADD_INFORMATION compiled


Table TEST created.


Table TEST dropped.
```



	USERNAME	EVENT	OBJECT_NAME	DATE_OF_EVENT
1	ALEX	CREATE	TEST	07-JAN-22
2	ALEX	DROP	TEST	07-JAN-22

13.

```
CREATE OR REPLACE PACKAGE pachet13 AS

   TYPE vector IS VARRAY(20) OF VARCHAR(20);
   TYPE tablou_imbricat IS TABLE OF NUMBER;
   PROCEDURE sortare_restaurante;
   PROCEDURE alimente;
   FUNCTION owner_price (price COST.VALUE%TYPE)
      RETURN VARCHAR;
   PROCEDURE pagini_meniu (nume CUSTOMER.NAME%TYPE);

END pachet13;
/
CREATE OR REPLACE PACKAGE BODY pachet13 AS

   PROCEDURE sortare_restaurante
   IS
      nume_restaurant vector := vector();
      capacitate tablou_imbricat := tablou_imbricat();
      gasit NUMBER;
      aux_number NUMBER;
      aux_char VARCHAR(20);
   BEGIN

      SELECT name
      BULK COLLECT INTO nume_restaurant
```

```
FROM restaurant;


SELECT capacity
BULK COLLECT INTO capacitate
FROM restaurant;


gasit := 0;


WHILE gasit = 0 LOOP


   gasit := 1;


   FOR j IN capacitate.FIRST..capacitate.LAST - 1 LOOP
      IF capacitate(j) > capacitate(j + 1) THEN
         aux_number := capacitate(j);
         capacitate(j) := capacitate(j + 1);
         capacitate(j + 1) := aux_number;


         aux_char := nume_restaurant(j);
         nume_restaurant(j) := nume_restaurant(j + 1);
         nume_restaurant(j + 1) := aux_char;


         gasit := 0;
      END IF;
   END LOOP;
   END LOOP;

   FOR j IN nume_restaurant.FIRST..nume_restaurant.LAST LOOP
      DBMS_OUTPUT.PUT_LINE (nume_restaurant(j) || ' ' || capacitate(j));
   END LOOP;

END sortare_restaurante;

PROCEDURE alimente
IS
   CURSOR c IS
      SELECT *
      FROM FOOD_ITEM
      WHERE price >= 25;
BEGIN

   FOR aliment IN c LOOP
      DBMS_OUTPUT.PUT_LINE(aliment.name || ' ' || aliment.price);
   END LOOP;

END alimente;

FUNCTION owner_price (price COST.VALUE%TYPE)
RETURN VARCHAR IS
   nume OWNER.LAST_NAME%TYPE;
BEGIN
```

```
    SELECT o.LAST_NAME INTO nume
    FROM RESTAURANT r JOIN OWNER o ON (o.OWNER_ID = r.OWNER_ID)
             JOIN COST c ON (c.RESTAURANT_ID = r.RESTAURANT_ID)
    WHERE c.VALUE = price;

    RETURN nume;

    EXCEPTION
      WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000, 'Nu exista niciun produs cu pretul
indicat!');
      WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20001, 'Exista mai multe produse cu
pretul indicat!');
      WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');

  END owner_price;

  PROCEDURE pagini_meniu (nume CUSTOMER.NAME%TYPE)
  IS
    numar_pagini MENU.MENU_ID%TYPE;

  BEGIN
    SELECT m.PAGE_NUMBER INTO numar_pagini
    FROM CUSTOMER c JOIN ORDER_DATA od ON (c.CUSTOMER_ID =
od.CUSTOMER_ID)
```

```
                JOIN ORDER_ITEM oi ON (od.ORDER_ID = oi.ORDER_ID)
                JOIN FOOD_ITEM f   ON (f.ITEM_ID = oi.ITEM_ID)
                JOIN MENU m        ON (m.MENU_ID  = f.MENU_ID)
        WHERE c.NAME = nume;

        DBMS_OUTPUT.PUT_LINE('Meniul costa ' || numar_pagini || ' lei.');

        EXCEPTION
          WHEN NO_DATA_FOUND THEN
            RAISE_APPLICATION_ERROR(-20000, 'Nu exista client cu acest nume
sau clientul nu a efectuat o comanda!');
          WHEN TOO_MANY_ROWS THEN
            RAISE_APPLICATION_ERROR(-20001, 'Exista mai multi clienti cu acest
nume!');
          WHEN OTHERS THEN
            RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');

    END pagini_meniu;

END pachet13;
```

14.

```
CREATE OR REPLACE PACKAGE pachet14 AS

    TYPE chelneri IS RECORD --memoreaza numele unui restaurant si numarul de
chelneri de la acel restaurant
    (
        nume RESTAURANT.NAME%TYPE,

        numar NUMBER

    );


    TYPE restaurante IS RECORD --memoreaza numele unui proprietar si numarul
de restaurante pe care il detine

    (

        nume OWNER.LAST_NAME%TYPE,

        numar_rest NUMBER

    );


    TYPE vector IS VARRAY(20) OF restaurante;

    TYPE vector_id IS VARRAY(20) OF RESTAURANT.RESTAURANT_ID%TYPE;


    FUNCTION numar_chelneri( v_id RESTAURANT.RESTAURANT_ID%TYPE) --
primeste ca parametru id-ul unui restaurant si returneaza numarul de chelneri

        RETURN chelneri;

    FUNCTION maxim_restaurante --returneaza numarul maxim de restaurante
detinut de un proprietar

        RETURN NUMBER;

    PROCEDURE maxim_proprietari; --afiseaza proprietarii cu numar maxim de
restaurante

    PROCEDURE suprapopulat; --afiseaza dintre restaurantele cu un singur
chelner pe cel cu capacitatea maxima(cel mai suprapopulat)


END pachet14;

/

CREATE OR REPLACE PACKAGE BODY pachet14 AS
```

```
FUNCTION numar_chelneri( v_id RESTAURANT.RESTAURANT_ID%TYPE)
RETURN chelneri IS
   v_chelner chelneri;
   v_numar NUMBER;
BEGIN
   SELECT COUNT(*) INTO v_chelner.numar
   FROM WAITER
   GROUP BY RESTAURANT_ID
   HAVING RESTAURANT_ID = v_id;

   SELECT NAME INTO v_chelner.nume
   FROM RESTAURANT
   WHERE RESTAURANT_ID = v_id;

   RETURN v_chelner;

END numar_chelneri;

FUNCTION maxim_restaurante
RETURN NUMBER IS
   v_maxim NUMBER;
BEGIN

   SELECT MAX(COUNT(*)) INTO v_maxim
   FROM RESTAURANT
```

```
    GROUP BY OWNER_ID;

    RETURN v_maxim;

END maxim_restaurante;

PROCEDURE maxim_proprietari
IS
    maxim_rest NUMBER;
    proprietari vector := vector();

BEGIN
    maxim_rest := maxim_restaurante;

    SELECT LAST_NAME, COUNT(*)
    BULK COLLECT INTO proprietari
    FROM OWNER o JOIN RESTAURANT r ON (o.OWNER_ID = R.OWNER_ID)
    GROUP BY LAST_NAME;

    FOR i IN proprietari.FIRST..proprietari.LAST LOOP

        IF proprietari(i).numar_rest = maxim_rest THEN
            DBMS_OUTPUT.PUT_LINE(proprietari(i).nume);
        END IF;

    END LOOP;
END maxim_proprietari;

PROCEDURE suprapopulat
IS
   rest vector_id := vector_id();
   val chelneri;
   nrmax NUMBER;
   rest_max RESTAURANT.NAME%TYPE;
   cap NUMBER;
BEGIN

   SELECT RESTAURANT_ID
   BULK COLLECT INTO rest
   FROM RESTAURANT;

   nrmax := -1;

   FOR i IN rest.FIRST..rest.LAST LOOP

      val := numar_chelneri(rest(i));

      IF val.numar = 1 THEN

         SELECT CAPACITY INTO cap
         FROM RESTAURANT
```

```
            WHERE RESTAURANT_ID = rest(i);

            IF cap > nrmax THEN
               rest_max := val.nume;
               nrmax := cap;
            END IF;

         END IF;

      END LOOP;

      DBMS_OUTPUT.PUT_LINE(rest_max);

   END suprapopulat;


END pachet14;
```