Proiect SGBD

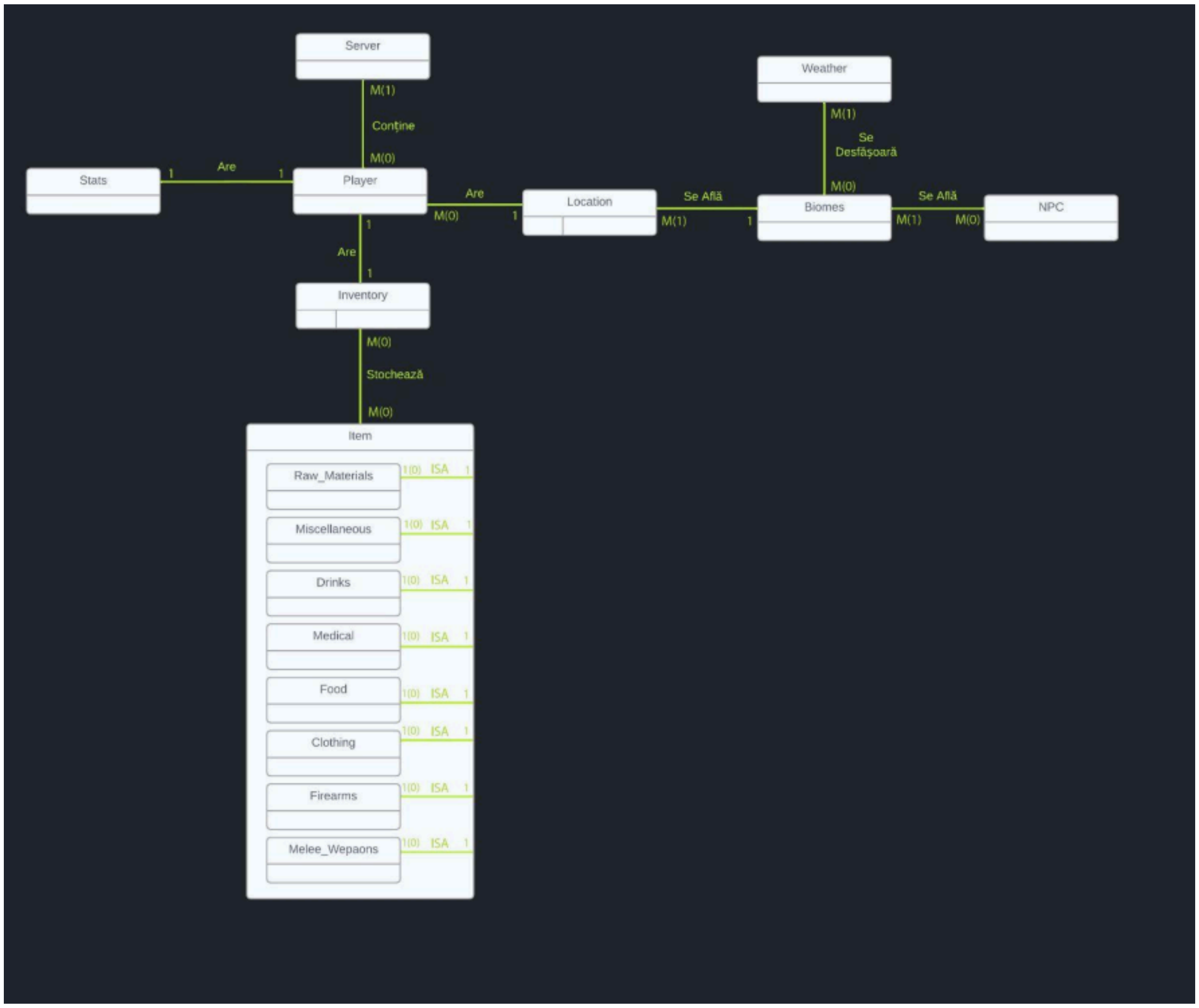
-Gestionarea unui joc survival-

Papuc Stefan-Eduard, grupa 242

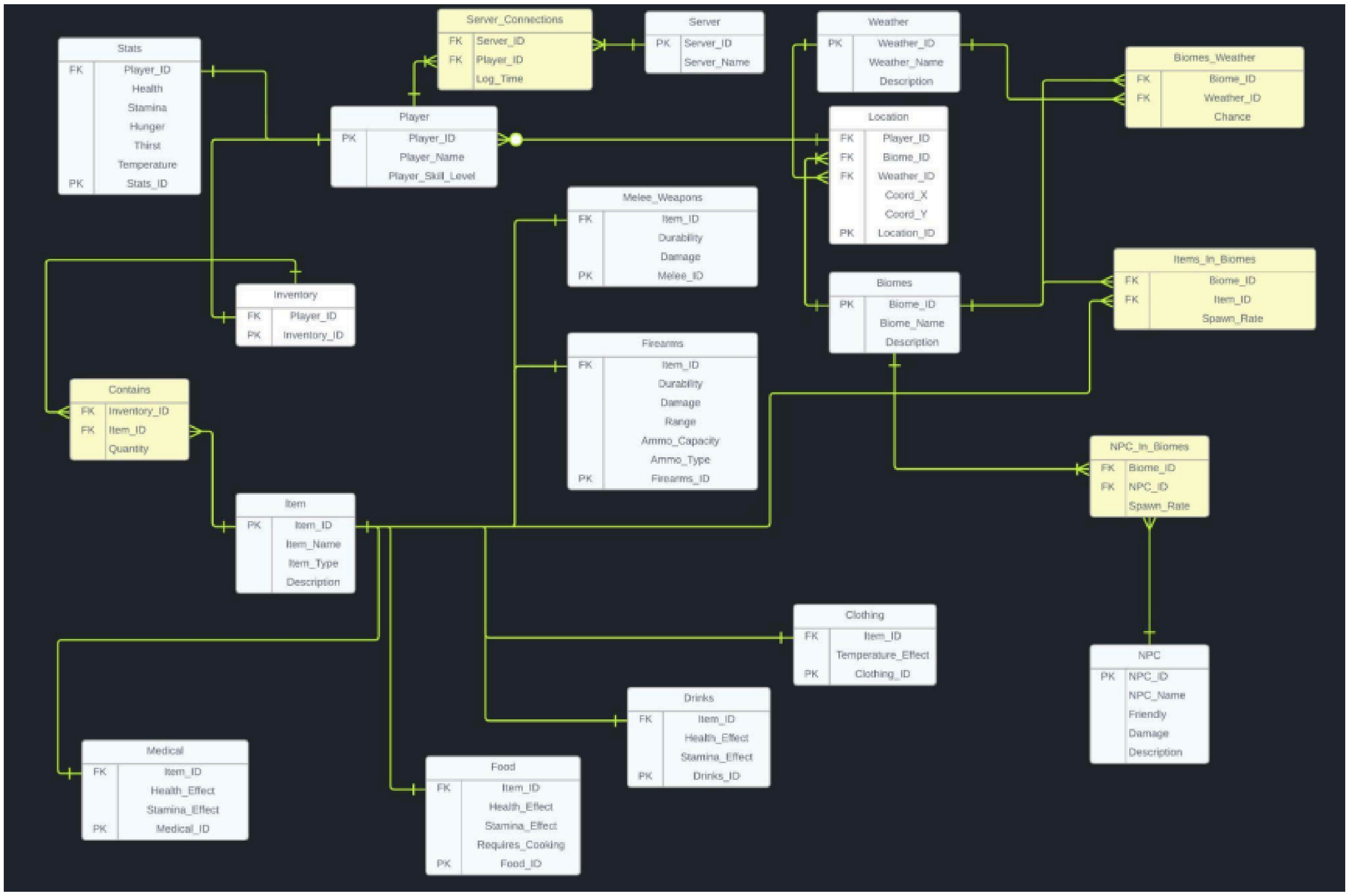
1. Prezentați pe scurt baza de date (utilitatea ei).

Baza de date va include informații despre resursele prezente în joc, precum apă, mâncare, materii prime (lemn, piatră, lut etc.), medicamente. De asemenea, vor fi stocate informații despre mediul în care se desfășoară jocul (tipuri de teren, condiții meteorologice, flora, fauna). Cele mai importante tabele vor conține date despre personajul jucătorului: inventarul, viața, nivelul de foame, de sete și temperatura corpului. Pentru a avea funcționalitatea de multiplayer, baza de date va stoca informații despre jucătorii de pe un anumit server, spre exemplu username-ul, inventarul și locația fiecărui jucător care alege modul multiplayer. Vremea actuală și biome-ul în care se află jucătorul vor determina ce NPC-uri (non-playable characters) poate întâlni acesta.

2. Realizați diagrama entitate-relație (ERD): entitățile, relațiile și atributele trebuie definite în limba română (vezi curs SGBD / model de diagrama ERD; nu se va accepta alt format).

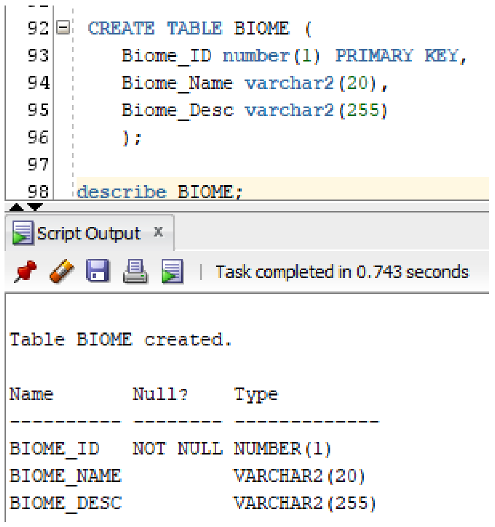


3. Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare: entitățile, relațiile și atributele trebuie definite în limba română.



4. Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, definind toate constrângerile de integritate necesare (chei primare, cheile externe etc).

```
CREATE TABLE BIOME (
Biome_ID number(1) PRIMARY KEY,
Biome_Name varchar2(20),
Biome_Desc varchar2(255)
);
```

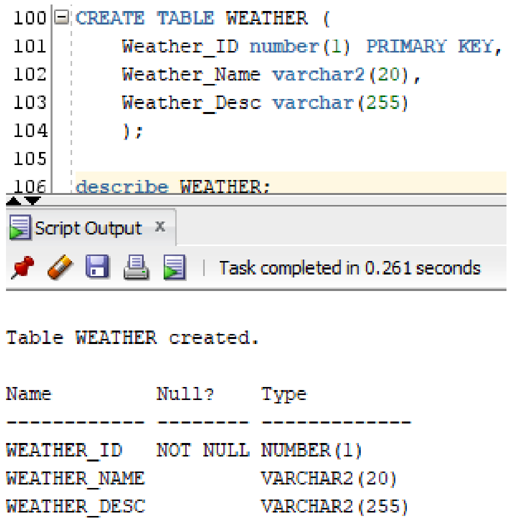


```
CREATE TABLE BIOME (
    Biome_ID number(1) PRIMARY KEY,
    Biome_Name varchar2(20),
    Biome_Desc varchar2(255)
    );

describe BIOME;
```

Script Output | Task completed in 0.743 seconds

```
Table BIOME created.

Name       Null?    Type
---------- -------- -------------
BIOME_ID   NOT NULL NUMBER(1)
BIOME_NAME          VARCHAR2(20)
BIOME_DESC          VARCHAR2(255)
```

```
CREATE TABLE WEATHER (
Weather_ID number(1) PRIMARY KEY,
Weather_Name varchar2(20),
Weather_Desc varchar(255)
);
```

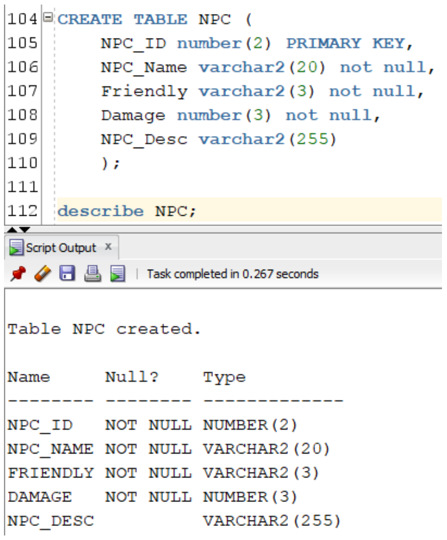


```
CREATE TABLE WEATHER (
    Weather_ID number(1) PRIMARY KEY,
    Weather_Name varchar2(20),
    Weather_Desc varchar(255)
    );

describe WEATHER;
```

Script Output | Task completed in 0.261 seconds

```
Table WEATHER created.

Name         Null?    Type
------------ -------- -------------
WEATHER_ID   NOT NULL NUMBER(1)
WEATHER_NAME          VARCHAR2(20)
WEATHER_DESC          VARCHAR2(255)
```

```
CREATE TABLE NPC (
NPC_ID number(2) PRIMARY KEY,
NPC_Name varchar2(20) not null,
Friendly varchar2(3) not null,
Damage number(3) not null,
NPC_Desc varchar2(255)
);
```



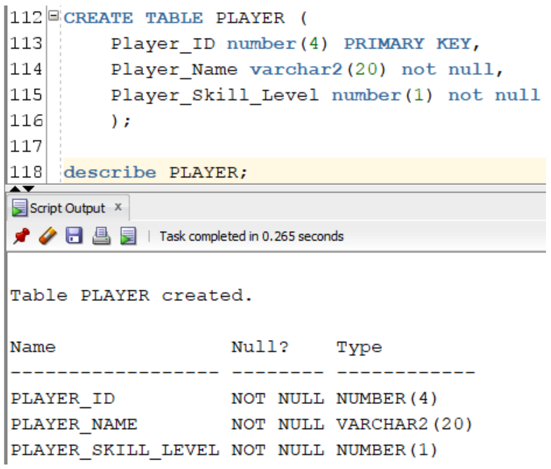
```
CREATE TABLE NPC (
    NPC_ID number(2) PRIMARY KEY,
    NPC_Name varchar2(20) not null,
    Friendly varchar2(3) not null,
    Damage number(3) not null,
    NPC_Desc varchar2(255)
    );

describe NPC;
```

Script Output ×

Task completed in 0.267 seconds

```
Table NPC created.

Name     Null?    Type
-------- -------- -------------
NPC_ID   NOT NULL NUMBER(2)
NPC_NAME NOT NULL VARCHAR2(20)
FRIENDLY NOT NULL VARCHAR2(3)
DAMAGE   NOT NULL NUMBER(3)
NPC_DESC          VARCHAR2(255)
```

```
CREATE TABLE PLAYER (
Player_ID number(4) PRIMARY KEY,
Player_Name varchar2(20) not null,
Player_Skill_Level number(1) not null
);
```



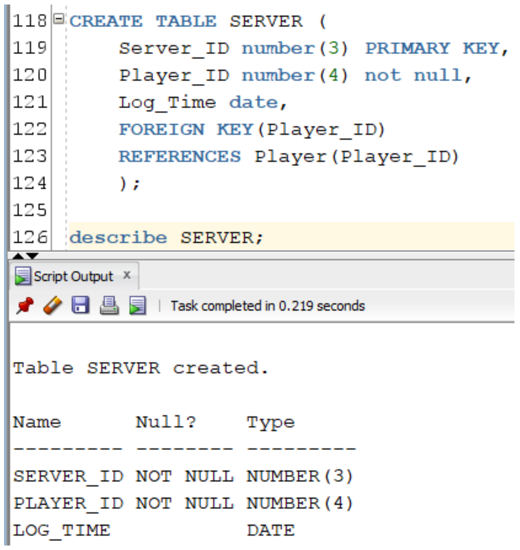
```
CREATE TABLE PLAYER (
    Player_ID number(4) PRIMARY KEY,
    Player_Name varchar2(20) not null,
    Player_Skill_Level number(1) not null
    );

describe PLAYER;
```

Script Output ×

Task completed in 0.265 seconds

```
Table PLAYER created.

Name               Null?    Type
------------------ -------- ------------
PLAYER_ID          NOT NULL NUMBER(4)
PLAYER_NAME        NOT NULL VARCHAR2(20)
PLAYER_SKILL_LEVEL NOT NULL NUMBER(1)
```

```
CREATE TABLE SERVER (
Server_ID number(3) PRIMARY KEY,
Player_ID number(4) not null,
Log_Time date,
FOREIGN KEY(Player_ID)
REFERENCES Player(Player_ID)
);
```



```
CREATE TABLE SERVER (
    Server_ID number(3) PRIMARY KEY,
    Player_ID number(4) not null,
    Log_Time date,
    FOREIGN KEY(Player_ID)
    REFERENCES Player(Player_ID)
    );

describe SERVER;
```

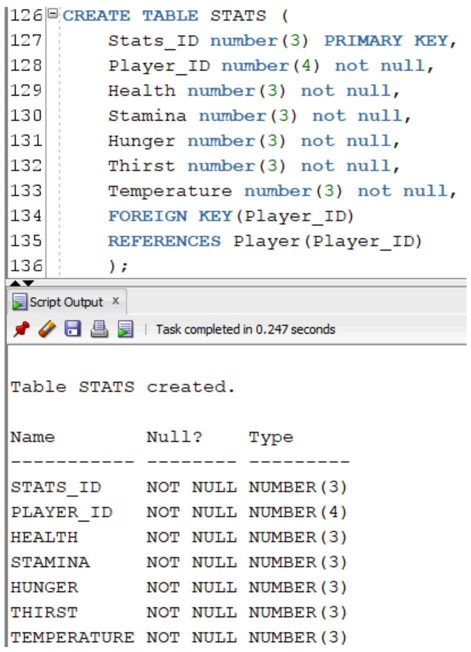
Script Output ×

Task completed in 0.219 seconds

```
Table SERVER created.

Name      Null?    Type
--------- -------- ---------
SERVER_ID NOT NULL NUMBER(3)
PLAYER_ID NOT NULL NUMBER(4)
LOG_TIME           DATE
```

```
ALTER TABLE SERVER (
DROP Player_ID,
DROP Log_Time,
ADD Server_Name varchar2(20)
);
```

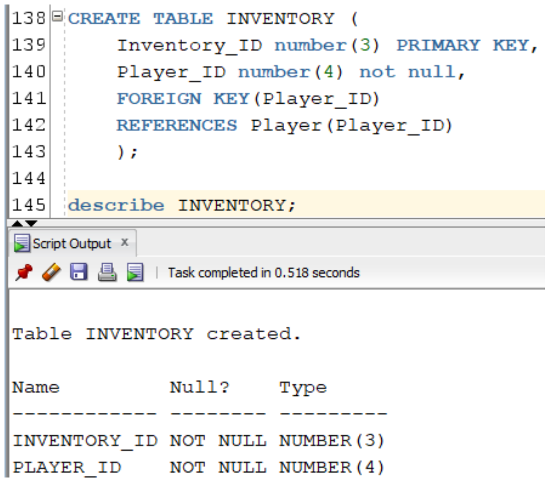
```
CREATE TABLE STATS (
Stats_ID number(3) PRIMARY KEY,
Player_ID number(4) not null,
Health number(3) not null,
Stamina number(3) not null,
Hunger number(3) not null,
Thirst number(3) not null,
Temperature number(3) not null,
FOREIGN KEY(Player_ID)
REFERENCES Player(Player_ID)
);
```



```
CREATE TABLE STATS (
    Stats_ID number(3) PRIMARY KEY,
    Player_ID number(4) not null,
    Health number(3) not null,
    Stamina number(3) not null,
    Hunger number(3) not null,
    Thirst number(3) not null,
    Temperature number(3) not null,
    FOREIGN KEY(Player_ID)
    REFERENCES Player(Player_ID)
    );
```

```
Table STATS created.

Name        Null?    Type
----------- -------- ---------
STATS_ID    NOT NULL NUMBER(3)
PLAYER_ID   NOT NULL NUMBER(4)
HEALTH      NOT NULL NUMBER(3)
STAMINA     NOT NULL NUMBER(3)
HUNGER      NOT NULL NUMBER(3)
THIRST      NOT NULL NUMBER(3)
TEMPERATURE NOT NULL NUMBER(3)
```

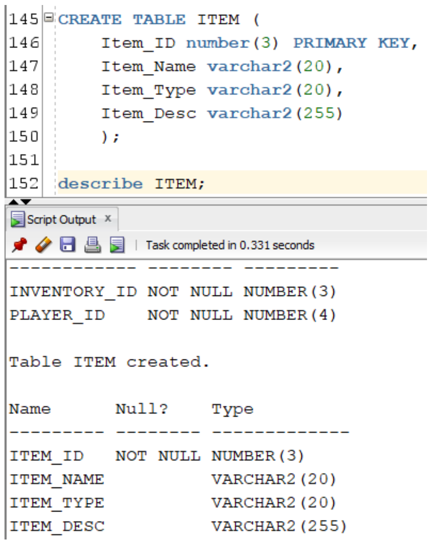
```
CREATE TABLE INVENTORY (
Inventory_ID number(3) PRIMARY KEY,
Player_ID number(4) not null,
FOREIGN KEY(Player_ID)
REFERENCES Player(Player_ID)
);
```



```
CREATE TABLE INVENTORY (
    Inventory_ID number(3) PRIMARY KEY,
    Player_ID number(4) not null,
    FOREIGN KEY(Player_ID)
    REFERENCES Player(Player_ID)
    );

describe INVENTORY;
```

```
Table INVENTORY created.

Name         Null?    Type
------------ -------- ---------
INVENTORY_ID NOT NULL NUMBER(3)
PLAYER_ID    NOT NULL NUMBER(4)
```

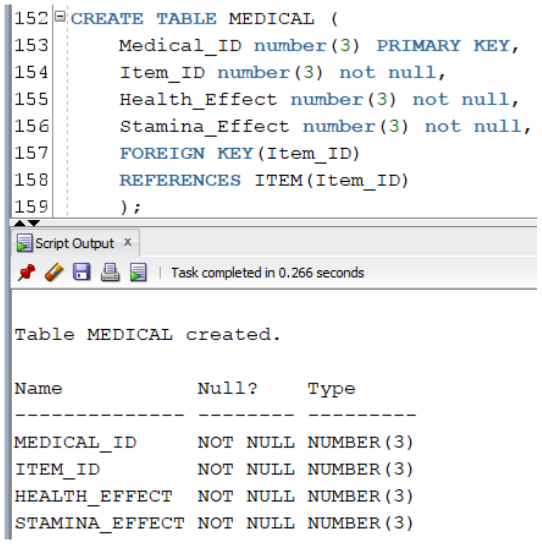
```
CREATE TABLE ITEM (
Item_ID number(3) PRIMARY KEY,
Item_Name varchar2(20),
Item_Type varchar2(20),
Item_Desc varchar2(255)
);
```



```
CREATE TABLE ITEM (
    Item_ID number(3) PRIMARY KEY,
    Item_Name varchar2(20),
    Item_Type varchar2(20),
    Item_Desc varchar2(255)
    );

describe ITEM;
```

```
Task completed in 0.331 seconds
------------ -------- ---------
INVENTORY_ID NOT NULL NUMBER(3)
PLAYER_ID    NOT NULL NUMBER(4)

Table ITEM created.

Name      Null?    Type
--------- -------- -------------
ITEM_ID   NOT NULL NUMBER(3)
ITEM_NAME          VARCHAR2(20)
ITEM_TYPE          VARCHAR2(20)
ITEM_DESC          VARCHAR2(255)
```

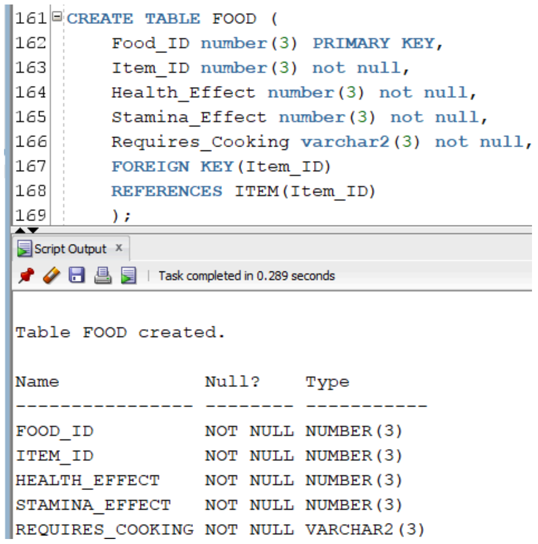
```
CREATE TABLE MEDICAL (
Medical_ID number(3) PRIMARY KEY,
Item_ID number(3) not null,
Health_Effect number(3) not null,
Stamina_Effect number(3) not null,
FOREIGN KEY(Item_ID)
REFERENCES ITEM(Item_ID)
);
```



```
CREATE TABLE MEDICAL (
    Medical_ID number(3) PRIMARY KEY,
    Item_ID number(3) not null,
    Health_Effect number(3) not null,
    Stamina_Effect number(3) not null,
    FOREIGN KEY(Item_ID)
    REFERENCES ITEM(Item_ID)
    );
```

```
Task completed in 0.266 seconds

Table MEDICAL created.

Name           Null?    Type
-------------- -------- ---------
MEDICAL_ID     NOT NULL NUMBER(3)
ITEM_ID        NOT NULL NUMBER(3)
HEALTH_EFFECT  NOT NULL NUMBER(3)
STAMINA_EFFECT NOT NULL NUMBER(3)
```

```
CREATE TABLE FOOD (
Food_ID number(3) PRIMARY KEY,
Item_ID number(3) not null,
Health_Effect number(3) not null,
Stamina_Effect number(3) not null,
Requires_Cooking varchar2(3) not null,
FOREIGN KEY(Item_ID)
```

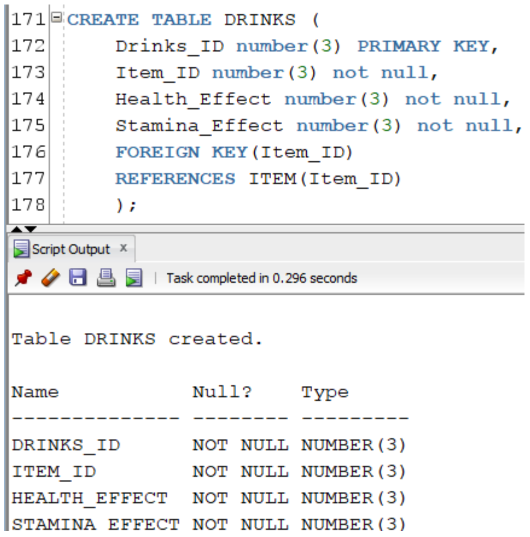


```
CREATE TABLE FOOD (
    Food_ID number(3) PRIMARY KEY,
    Item_ID number(3) not null,
    Health_Effect number(3) not null,
    Stamina_Effect number(3) not null,
    Requires_Cooking varchar2(3) not null,
    FOREIGN KEY(Item_ID)
    REFERENCES ITEM(Item_ID)
    );
```

```
Task completed in 0.289 seconds

Table FOOD created.

Name             Null?    Type
---------------- -------- -----------
FOOD_ID          NOT NULL NUMBER(3)
ITEM_ID          NOT NULL NUMBER(3)
HEALTH_EFFECT    NOT NULL NUMBER(3)
STAMINA_EFFECT   NOT NULL NUMBER(3)
REQUIRES_COOKING NOT NULL VARCHAR2(3)
```

```
REFERENCES ITEM(Item_ID)
);
```

```
CREATE TABLE DRINKS (
Drinks_ID number(3) PRIMARY KEY,
Item_ID number(3) not null,
Health_Effect number(3) not null,
Stamina_Effect number(3) not null,
FOREIGN KEY(Item_ID)
REFERENCES ITEM(Item_ID)
);
```



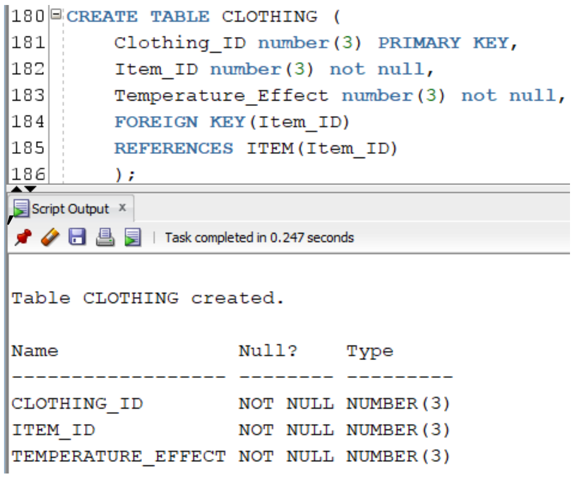
```
CREATE TABLE DRINKS (
    Drinks_ID number(3) PRIMARY KEY,
    Item_ID number(3) not null,
    Health_Effect number(3) not null,
    Stamina_Effect number(3) not null,
    FOREIGN KEY(Item_ID)
    REFERENCES ITEM(Item_ID)
    );
```

Script Output ×

Task completed in 0.296 seconds

```
Table DRINKS created.

Name           Null?    Type
-------------- -------- ---------
DRINKS_ID      NOT NULL NUMBER(3)
ITEM_ID        NOT NULL NUMBER(3)
HEALTH_EFFECT  NOT NULL NUMBER(3)
STAMINA_EFFECT NOT NULL NUMBER(3)
```

```
CREATE TABLE CLOTHING (
Clothing_ID number(3) PRIMARY KEY,
Item_ID number(3) not null,
Temperature_Effect number(3) not null,
FOREIGN KEY(Item_ID)
REFERENCES ITEM(Item_ID)
);
```



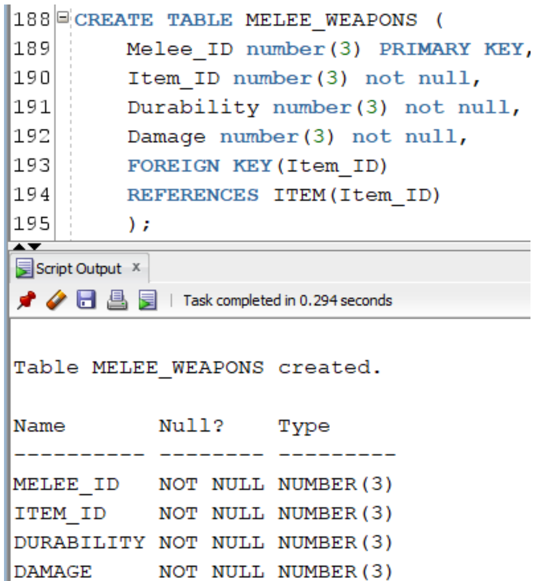
```
CREATE TABLE CLOTHING (
    Clothing_ID number(3) PRIMARY KEY,
    Item_ID number(3) not null,
    Temperature_Effect number(3) not null,
    FOREIGN KEY(Item_ID)
    REFERENCES ITEM(Item_ID)
    );
```

Script Output ×

Task completed in 0.247 seconds

```
Table CLOTHING created.

Name               Null?    Type
------------------ -------- ---------
CLOTHING_ID        NOT NULL NUMBER(3)
ITEM_ID            NOT NULL NUMBER(3)
TEMPERATURE_EFFECT NOT NULL NUMBER(3)
```

```
CREATE TABLE MELEE_WEAPONS (
Melee_ID number(3) PRIMARY KEY,
Item_ID number(3) not null,
Durability number(3) not null,
Damage number(3) not null,
FOREIGN KEY(Item_ID)
REFERENCES ITEM(Item_ID)
```



```
CREATE TABLE MELEE_WEAPONS (
    Melee_ID number(3) PRIMARY KEY,
    Item_ID number(3) not null,
    Durability number(3) not null,
    Damage number(3) not null,
    FOREIGN KEY(Item_ID)
    REFERENCES ITEM(Item_ID)
    );
```

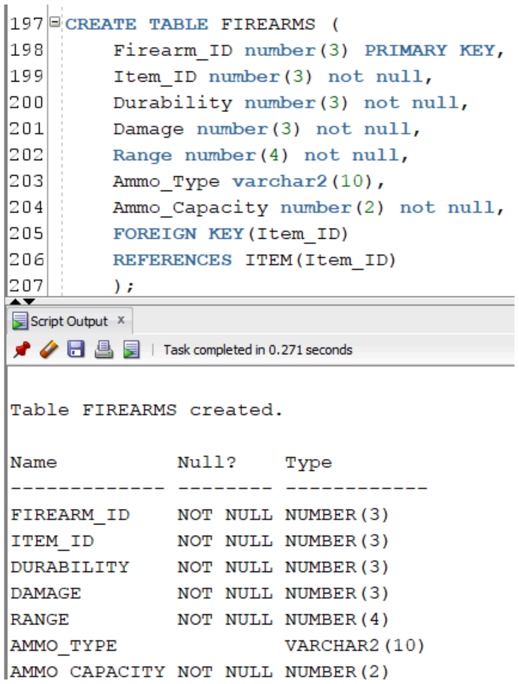
Script Output ×

Task completed in 0.294 seconds

```
Table MELEE_WEAPONS created.

Name       Null?    Type
---------- -------- ---------
MELEE_ID   NOT NULL NUMBER(3)
ITEM_ID    NOT NULL NUMBER(3)
DURABILITY NOT NULL NUMBER(3)
DAMAGE     NOT NULL NUMBER(3)
```

```
);
```

```
CREATE TABLE FIREARMS (
Firearm_ID number(3) PRIMARY KEY,
Item_ID number(3) not null,
Durability number(3) not null,
Damage number(3) not null,
Range number(4) not null,
Ammo_Type varchar2(10),
Ammo_Capacity number(2) not null,
FOREIGN KEY(Item_ID)
REFERENCES ITEM(Item_ID)
);
```



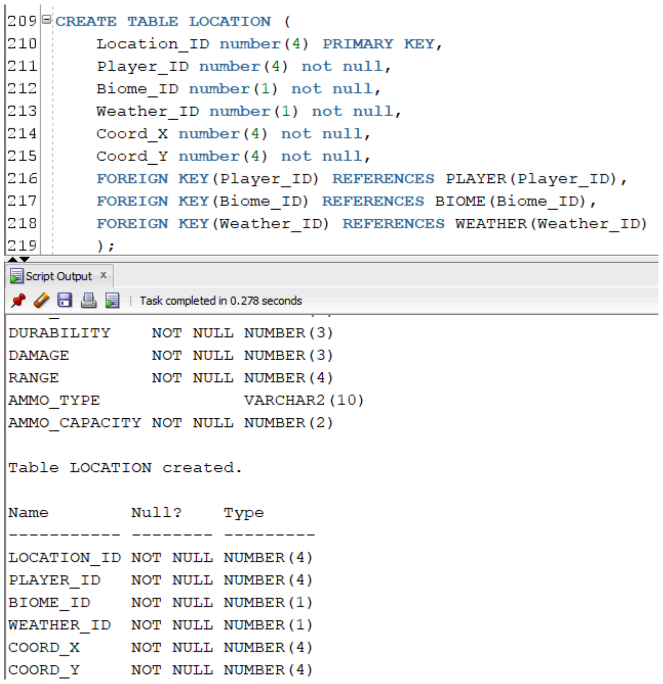
```
CREATE TABLE FIREARMS (
    Firearm_ID number(3) PRIMARY KEY,
    Item_ID number(3) not null,
    Durability number(3) not null,
    Damage number(3) not null,
    Range number(4) not null,
    Ammo_Type varchar2(10),
    Ammo_Capacity number(2) not null,
    FOREIGN KEY(Item_ID)
    REFERENCES ITEM(Item_ID)
    );
```

Script Output ×

Task completed in 0.271 seconds

```
Table FIREARMS created.

Name          Null?    Type
------------- -------- ------------
FIREARM_ID    NOT NULL NUMBER(3)
ITEM_ID       NOT NULL NUMBER(3)
DURABILITY    NOT NULL NUMBER(3)
DAMAGE        NOT NULL NUMBER(3)
RANGE         NOT NULL NUMBER(4)
AMMO_TYPE              VARCHAR2(10)
AMMO_CAPACITY NOT NULL NUMBER(2)
```

```
CREATE TABLE LOCATION (
Location_ID number(4) PRIMARY KEY,
Player_ID number(4) not null,
Biome_ID number(1) not null,
Weather_ID number(1) not null,
Coord_X number(4) not null,
Coord_Y number(4) not null,
FOREIGN KEY(Player_ID)
REFERENCES PLAYER(Player_ID),
FOREIGN KEY(Biome_ID)
REFERENCES BIOME(Biome_ID),
FOREIGN KEY(Weather_ID)
REFERENCES WEATHER(Weather_ID)
);
```



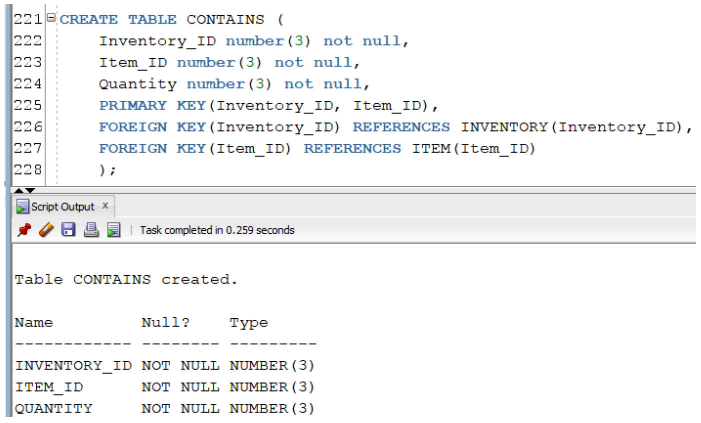
```
CREATE TABLE LOCATION (
    Location_ID number(4) PRIMARY KEY,
    Player_ID number(4) not null,
    Biome_ID number(1) not null,
    Weather_ID number(1) not null,
    Coord_X number(4) not null,
    Coord_Y number(4) not null,
    FOREIGN KEY(Player_ID) REFERENCES PLAYER(Player_ID),
    FOREIGN KEY(Biome_ID) REFERENCES BIOME(Biome_ID),
    FOREIGN KEY(Weather_ID) REFERENCES WEATHER(Weather_ID)
    );
```

Script Output ×

Task completed in 0.278 seconds

```
DURABILITY    NOT NULL NUMBER(3)
DAMAGE        NOT NULL NUMBER(3)
RANGE         NOT NULL NUMBER(4)
AMMO_TYPE              VARCHAR2(10)
AMMO_CAPACITY NOT NULL NUMBER(2)

Table LOCATION created.

Name        Null?    Type
----------- -------- ---------
LOCATION_ID NOT NULL NUMBER(4)
PLAYER_ID   NOT NULL NUMBER(4)
BIOME_ID    NOT NULL NUMBER(1)
WEATHER_ID  NOT NULL NUMBER(1)
COORD_X     NOT NULL NUMBER(4)
COORD_Y     NOT NULL NUMBER(4)
```

```
CREATE TABLE CONTAINS (
Inventory_ID number(3) not null,
Item_ID number(3) not null,
Quantity number(3) not null,
PRIMARY KEY(Inventory_ID, Item_ID),
FOREIGN KEY(Inventory_ID)
REFERENCES INVENTORY(Inventory_ID),
FOREIGN KEY(Item_ID)
REFERENCES ITEM(Item_ID)
);
```



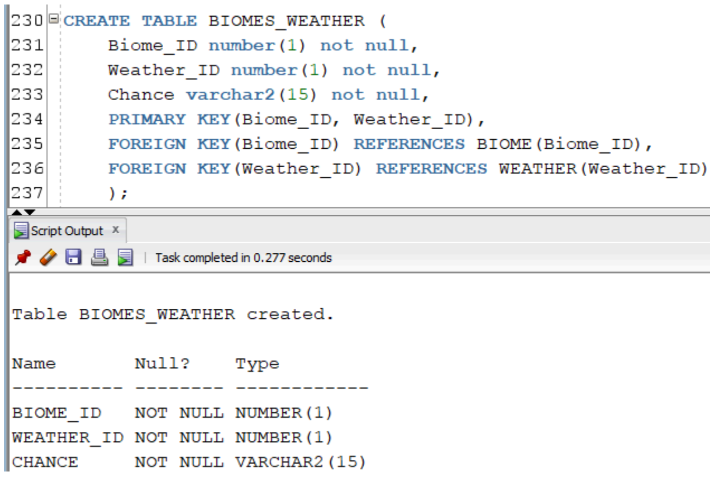
```
CREATE TABLE CONTAINS (
    Inventory_ID number(3) not null,
    Item_ID number(3) not null,
    Quantity number(3) not null,
    PRIMARY KEY(Inventory_ID, Item_ID),
    FOREIGN KEY(Inventory_ID) REFERENCES INVENTORY(Inventory_ID),
    FOREIGN KEY(Item_ID) REFERENCES ITEM(Item_ID)
    );
```

Script Output ×

Task completed in 0.259 seconds

```
Table CONTAINS created.

Name         Null?    Type
------------ -------- ---------
INVENTORY_ID NOT NULL NUMBER(3)
ITEM_ID      NOT NULL NUMBER(3)
QUANTITY     NOT NULL NUMBER(3)
```

```
CREATE TABLE BIOMES_WEATHER (
Biome_ID number(1) not null,
Weather_ID number(1) not null,
Chance varchar2(15) not null,
PRIMARY KEY(Biome_ID, Weather_ID),
FOREIGN KEY(Biome_ID)
REFERENCES BIOME(Biome_ID),
FOREIGN KEY(Weather_ID)
REFERENCES WEATHER(Weather_ID)
);
```



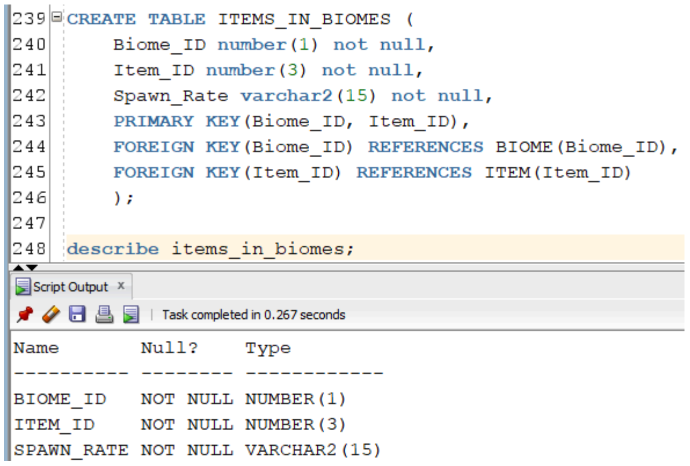
```
CREATE TABLE BIOMES_WEATHER (
    Biome_ID number(1) not null,
    Weather_ID number(1) not null,
    Chance varchar2(15) not null,
    PRIMARY KEY(Biome_ID, Weather_ID),
    FOREIGN KEY(Biome_ID) REFERENCES BIOME(Biome_ID),
    FOREIGN KEY(Weather_ID) REFERENCES WEATHER(Weather_ID)
    );
```

Script Output ×

Task completed in 0.277 seconds

```
Table BIOMES_WEATHER created.

Name       Null?    Type
---------- -------- ------------
BIOME_ID   NOT NULL NUMBER(1)
WEATHER_ID NOT NULL NUMBER(1)
CHANCE     NOT NULL VARCHAR2(15)
```

```
CREATE TABLE ITEMS_IN_BIOMES (
Biome_ID number(1) not null,
Item_ID number(3) not null,
Spawn_Rate varchar2(15) not null,
PRIMARY KEY(Biome_ID, Item_ID),
FOREIGN KEY(Biome_ID)
REFERENCES BIOME(Biome_ID),
FOREIGN KEY(Item_ID)
REFERENCES ITEM(Item_ID)
```



```
CREATE TABLE ITEMS_IN_BIOMES (
    Biome_ID number(1) not null,
    Item_ID number(3) not null,
    Spawn_Rate varchar2(15) not null,
    PRIMARY KEY(Biome_ID, Item_ID),
    FOREIGN KEY(Biome_ID) REFERENCES BIOME(Biome_ID),
    FOREIGN KEY(Item_ID) REFERENCES ITEM(Item_ID)
    );

describe items_in_biomes;
```

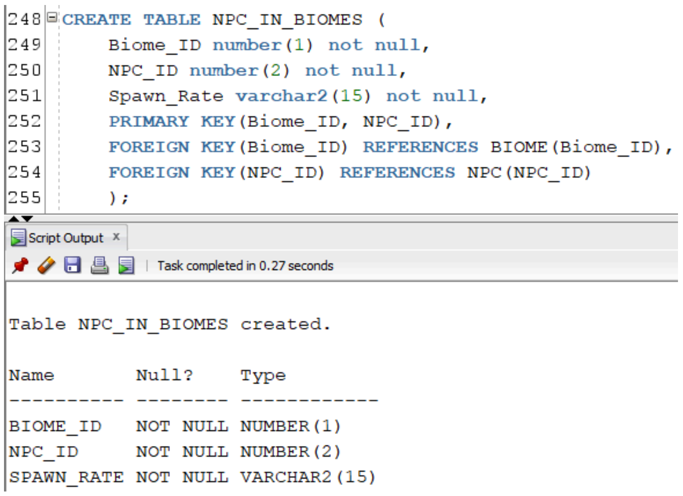
Script Output ×

Task completed in 0.267 seconds

```
Name       Null?    Type
---------- -------- ------------
BIOME_ID   NOT NULL NUMBER(1)
ITEM_ID    NOT NULL NUMBER(3)
SPAWN_RATE NOT NULL VARCHAR2(15)
```

```
);
```

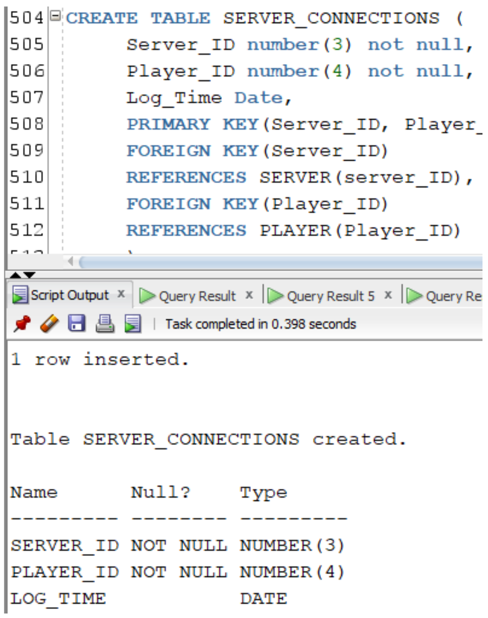
```
CREATE TABLE NPC_IN_BIOMES (
Biome_ID number(1) not null,
NPC_ID number(2) not null,
Spawn_Rate varchar2(15) not null,
PRIMARY KEY(Biome_ID, NPC_ID),
FOREIGN KEY(Biome_ID)
REFERENCES BIOME(Biome_ID),
FOREIGN KEY(NPC_ID
REFERENCES NPC(NPC_ID)
);
```



```
CREATE TABLE NPC_IN_BIOMES (
    Biome_ID number(1) not null,
    NPC_ID number(2) not null,
    Spawn_Rate varchar2(15) not null,
    PRIMARY KEY(Biome_ID, NPC_ID),
    FOREIGN KEY(Biome_ID) REFERENCES BIOME(Biome_ID),
    FOREIGN KEY(NPC_ID) REFERENCES NPC(NPC_ID)
    );
```

```
Table NPC_IN_BIOMES created.

Name       Null?    Type
---------- -------- ------------
BIOME_ID   NOT NULL NUMBER(1)
NPC_ID     NOT NULL NUMBER(2)
SPAWN_RATE NOT NULL VARCHAR2(15)
```

```
CREATE TABLE SERVER_CONNECTIONS (
Server_ID number(3) not null,
Player_ID number(4) not null,
Log_Time Date,
PRIMARY KEY(Server_ID, Player_ID),
FOREIGN KEY(Server_ID)
REFERENCES SERVER(server_ID),
FOREIGN KEY(Player_ID)
REFERENCES PLAYER(Player_ID)
);
```

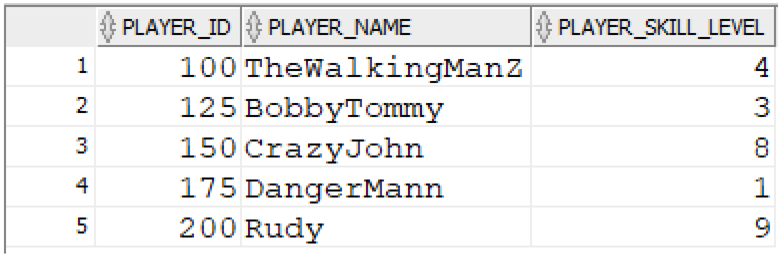


```
CREATE TABLE SERVER_CONNECTIONS (
    Server_ID number(3) not null,
    Player_ID number(4) not null,
    Log_Time Date,
    PRIMARY KEY(Server_ID, Player_
    FOREIGN KEY(Server_ID)
    REFERENCES SERVER(server_ID),
    FOREIGN KEY(Player_ID)
    REFERENCES PLAYER(Player_ID)
```

```
1 row inserted.


Table SERVER_CONNECTIONS created.

Name      Null?    Type
--------- -------- ---------
SERVER_ID NOT NULL NUMBER(3)
PLAYER_ID NOT NULL NUMBER(4)
LOG_TIME           DATE
```

5. Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă).

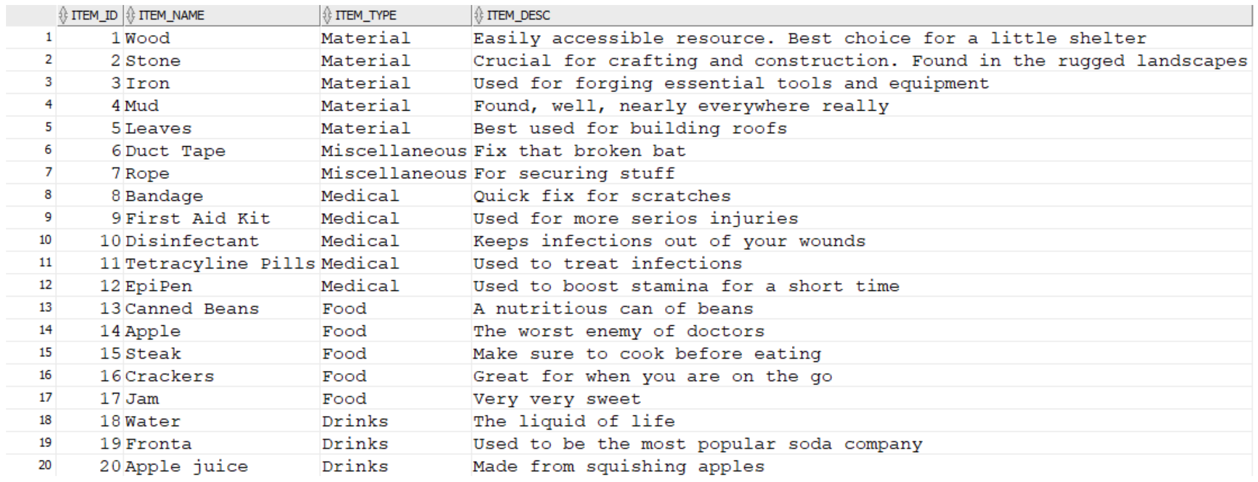
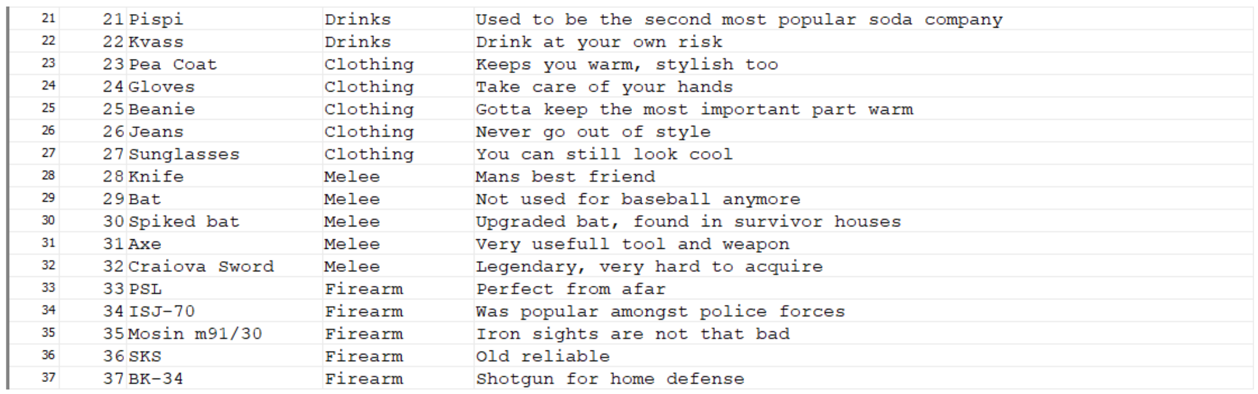
```
--player
INSERT INTO PLAYER VALUES(seq_player.nextval, 'TheWalkingManZ',
4);
INSERT INTO PLAYER VALUES(seq_player.nextval, 'BobbyTommy', 3);
INSERT INTO PLAYER VALUES(seq_player.nextval, 'CrazyJohn', 8);
INSERT INTO PLAYER VALUES(seq_player.nextval, 'DangerMann', 1);
INSERT INTO PLAYER VALUES(seq_player.nextval, 'Rudy', 9);
```



	PLAYER_ID	PLAYER_NAME	PLAYER_SKILL_LEVEL
1	100	TheWalkingManZ	4
2	125	BobbyTommy	3
3	150	CrazyJohn	8
4	175	DangerMann	1
5	200	Rudy	9

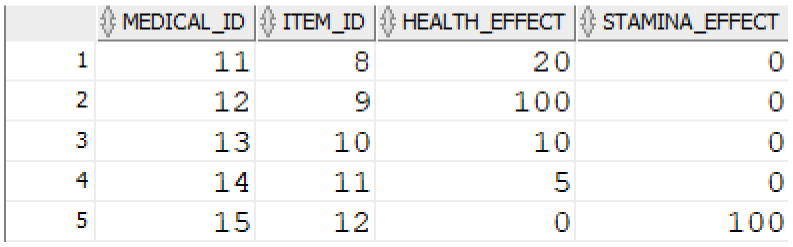
```
--item
INSERT INTO ITEM VALUES(seq_item.nextval, 'Wood', 'Material',
'Easily accessible resource. Best choice for a little shelter');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Stone', 'Material',
'Crucial for crafting and construction. Found in the rugged
landscapes');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Iron', 'Material',
'Used for forging essential tools and equipment');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Mud', 'Material',
'Found, well, nearly everywhere really');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Leaves', 'Material',
'Best used for building roofs');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Duct Tape',
'Miscellaneous', 'Fix that broken bat');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Rope',
'Miscellaneous', 'For securing stuff');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Bandage', 'Medical',
'Quick fix for scratches');
```

```
INSERT INTO ITEM VALUES(seq_item.nextval, 'First Aid Kit',
'Medical', 'Used for more serious injuries');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Disinfectant',
'Medical', 'Keeps infections out of your wounds');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Tetracyline Pills',
'Medical', 'Used to treat infections');
INSERT INTO ITEM VALUES(seq_item.nextval, 'EpiPen', 'Medical',
'Used to boost stamina for a short time');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Canned Beans',
'Food', 'A nutritious can of beans');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Apple', 'Food', 'The
worst enemy of doctors');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Steak', 'Food', 'Make
sure to cook before eating');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Crackers', 'Food',
'Great for when you are on the go');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Jam', 'Food', 'Very
very sweet');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Water', 'Drinks',
'The liquid of life');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Fronta', 'Drinks',
'Used to be the most popular soda company');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Apple juice',
'Drinks', 'Made from squishing apples');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Pispi', 'Drinks',
'Used to be the second most popular soda company');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Kvass', 'Drinks',
'Drink at your own risk');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Pea Coat',
'Clothing', 'Keeps you warm, stylish too');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Gloves', 'Clothing',
'Take care of your hands');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Beanie', 'Clothing',
'Gotta keep the most important part warm');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Jeans', 'Clothing',
'Never go out of style');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Sunglasses',
'Clothing', 'You can still look cool');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Knife', 'Melee',
'Mans best friend');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Bat', 'Melee', 'Not
used for baseball anymore');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Spiked bat', 'Melee',
'Upgraded bat, found in survivor houses');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Axe', 'Melee', 'Very
useful tool and weapon');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Craiova Sword',
'Melee', 'Legendary, very hard to acquire');
INSERT INTO ITEM VALUES(seq_item.nextval, 'PSL', 'Firearm',
'Perfect from afar');
INSERT INTO ITEM VALUES(seq_item.nextval, 'ISJ-70', 'Firearm',
'Was popular amongst police forces');
INSERT INTO ITEM VALUES(seq_item.nextval, 'Mosin m91/30',
'Firearm', 'Iron sights are not that bad');
INSERT INTO ITEM VALUES(seq_item.nextval, 'SKS', 'Firearm', 'Old
reliable');
INSERT INTO ITEM VALUES(seq_item.nextval, 'BK-34', 'Firearm',
'Shotgun for home defense');
```

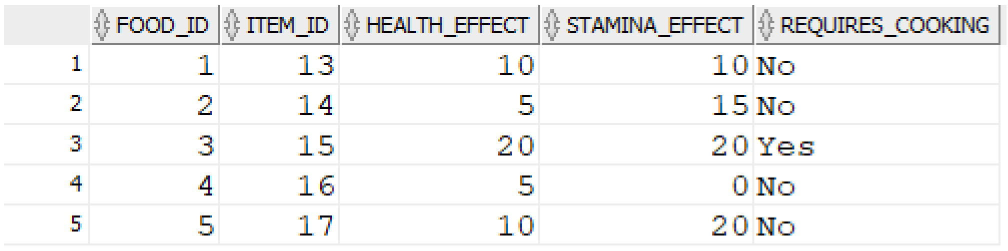
	ITEM_ID	ITEM_NAME	ITEM_TYPE	ITEM_DESC
1	1	Wood	Material	Easily accessible resource. Best choice for a little shelter
2	2	Stone	Material	Crucial for crafting and construction. Found in the rugged landscapes
3	3	Iron	Material	Used for forging essential tools and equipment
4	4	Mud	Material	Found, well, nearly everywhere really
5	5	Leaves	Material	Best used for building roofs
6	6	Duct Tape	Miscellaneous	Fix that broken bat
7	7	Rope	Miscellaneous	For securing stuff
8	8	Bandage	Medical	Quick fix for scratches
9	9	First Aid Kit	Medical	Used for more serios injuries
10	10	Disinfectant	Medical	Keeps infections out of your wounds
11	11	Tetracyline Pills	Medical	Used to treat infections
12	12	EpiPen	Medical	Used to boost stamina for a short time
13	13	Canned Beans	Food	A nutritious can of beans
14	14	Apple	Food	The worst enemy of doctors
15	15	Steak	Food	Make sure to cook before eating
16	16	Crackers	Food	Great for when you are on the go
17	17	Jam	Food	Very very sweet
18	18	Water	Drinks	The liquid of life
19	19	Fronta	Drinks	Used to be the most popular soda company
20	20	Apple juice	Drinks	Made from squishing apples
21	21	Pispi	Drinks	Used to be the second most popular soda company
22	22	Kvass	Drinks	Drink at your own risk
23	23	Pea Coat	Clothing	Keeps you warm, stylish too
24	24	Gloves	Clothing	Take care of your hands
25	25	Beanie	Clothing	Gotta keep the most important part warm
26	26	Jeans	Clothing	Never go out of style
27	27	Sunglasses	Clothing	You can still look cool
28	28	Knife	Melee	Mans best friend
29	29	Bat	Melee	Not used for baseball anymore
30	30	Spiked bat	Melee	Upgraded bat, found in survivor houses
31	31	Axe	Melee	Very usefull tool and weapon
32	32	Craiova Sword	Melee	Legendary, very hard to acquire
33	33	PSL	Firearm	Perfect from afar
34	34	ISJ-70	Firearm	Was popular amongst police forces
35	35	Mosin m91/30	Firearm	Iron sights are not that bad
36	36	SKS	Firearm	Old reliable
37	37	BK-34	Firearm	Shotgun for home defense

```
--medical
INSERT INTO MEDICAL VALUES(seq_medical.nextval, 8, 20, 0);
INSERT INTO MEDICAL VALUES(seq_medical.nextval, 9, 100, 0);
INSERT INTO MEDICAL VALUES(seq_medical.nextval, 10, 10, 0);
INSERT INTO MEDICAL VALUES(seq_medical.nextval, 11, 5, 0);
INSERT INTO MEDICAL VALUES(seq_medical.nextval, 12, 0, 100);
```



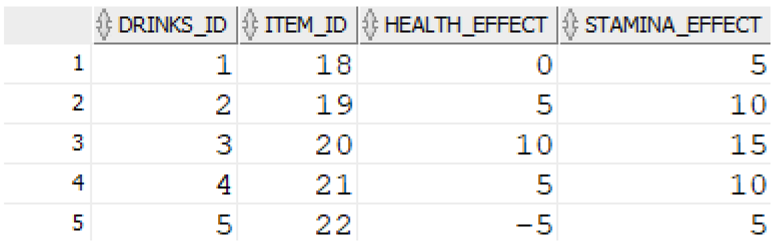
	MEDICAL_ID	ITEM_ID	HEALTH_EFFECT	STAMINA_EFFECT
1	11	8	20	0
2	12	9	100	0
3	13	10	10	0
4	14	11	5	0
5	15	12	0	100

```
--food
INSERT INTO FOOD VALUES(seq_food.nextval, 13, 10, 10, 'No');
INSERT INTO FOOD VALUES(seq_food.nextval, 14, 5, 15, 'No');
INSERT INTO FOOD VALUES(seq_food.nextval, 15, 20, 20, 'Yes');
INSERT INTO FOOD VALUES(seq_food.nextval, 16, 5, 0, 'No');
INSERT INTO FOOD VALUES(seq_food.nextval, 17, 10, 20, 'No');
```



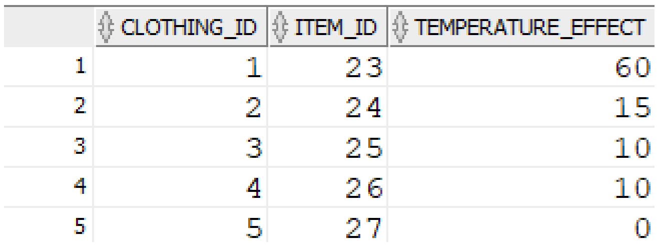
	FOOD_ID	ITEM_ID	HEALTH_EFFECT	STAMINA_EFFECT	REQUIRES_COOKING
1	1	13	10	10	No
2	2	14	5	15	No
3	3	15	20	20	Yes
4	4	16	5	0	No
5	5	17	10	20	No

```
--drinks
INSERT INTO DRINKS VALUES(seq_drinks.nextval, 18, 0, 5);
INSERT INTO DRINKS VALUES(seq_drinks.nextval, 19, 5, 10);
INSERT INTO DRINKS VALUES(seq_drinks.nextval, 20, 10, 15);
INSERT INTO DRINKS VALUES(seq_drinks.nextval, 21, 5, 10);
INSERT INTO DRINKS VALUES(seq_drinks.nextval, 22, -5, 5);
```



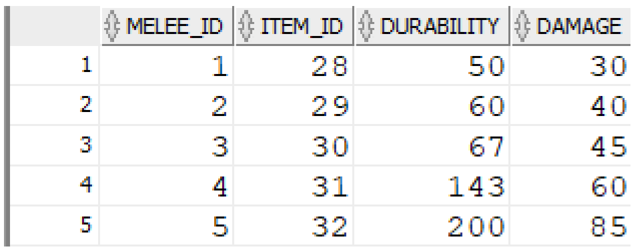
	DRINKS_ID	ITEM_ID	HEALTH_EFFECT	STAMINA_EFFECT
1	1	18	0	5
2	2	19	5	10
3	3	20	10	15
4	4	21	5	10
5	5	22	-5	5

```
--clothing
INSERT INTO CLOTHING VALUES(seq_clothing.nextval, 23, 60);
INSERT INTO CLOTHING VALUES(seq_clothing.nextval, 24, 15);
INSERT INTO CLOTHING VALUES(seq_clothing.nextval, 25, 10);
INSERT INTO CLOTHING VALUES(seq_clothing.nextval, 26, 10);
INSERT INTO CLOTHING VALUES(seq_clothing.nextval, 27, 0);
```



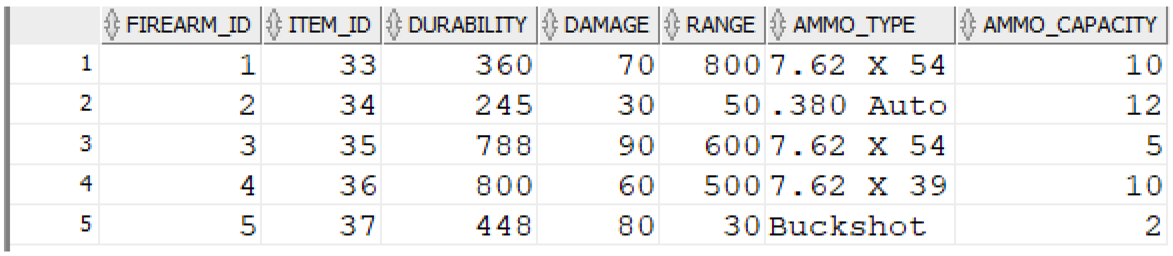
	CLOTHING_ID	ITEM_ID	TEMPERATURE_EFFECT
1	1	23	60
2	2	24	15
3	3	25	10
4	4	26	10
5	5	27	0

```
--melee
INSERT INTO MELEE_WEAPONS VALUES(seq_melee.nextval, 28, 50, 20);
INSERT INTO MELEE_WEAPONS VALUES(seq_melee.nextval, 29, 60, 30);
INSERT INTO MELEE_WEAPONS VALUES(seq_melee.nextval, 30, 67, 45);
INSERT INTO MELEE_WEAPONS VALUES(seq_melee.nextval, 31, 143,
60);
INSERT INTO MELEE_WEAPONS VALUES(seq_melee.nextval, 32, 200,
85);
```



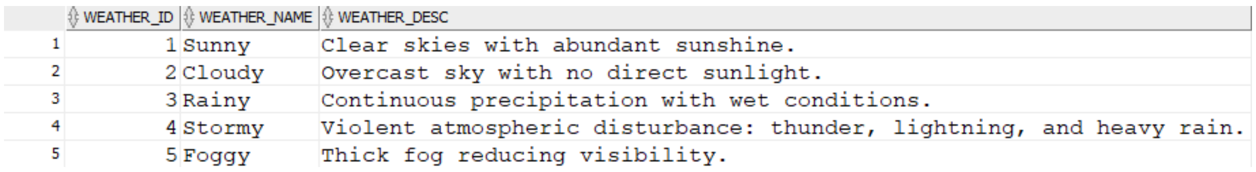
	MELEE_ID	ITEM_ID	DURABILITY	DAMAGE
1	1	28	50	30
2	2	29	60	40
3	3	30	67	45
4	4	31	143	60
5	5	32	200	85

```
--firearms
INSERT INTO FIREARMS VALUES(seq_firearms.nextval, 33, 360, 70,
800, '7.62 X 54', 10);
INSERT INTO FIREARMS VALUES(seq_firearms.nextval, 34, 245, 30,
50, '.380 Auto', 12);
INSERT INTO FIREARMS VALUES(seq_firearms.nextval, 35, 788, 90,
600, '7.62 X 54', 5);
INSERT INTO FIREARMS VALUES(seq_firearms.nextval, 36, 800, 60,
500, '7.62 X 39', 10);
INSERT INTO FIREARMS VALUES(seq_firearms.nextval, 37, 448, 80,
30, 'Buckshot', 2);
```



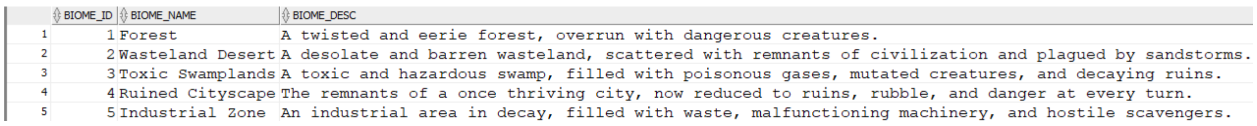
	FIREARM_ID	ITEM_ID	DURABILITY	DAMAGE	RANGE	AMMO_TYPE	AMMO_CAPACITY
1	1	33	360	70	800	7.62 X 54	10
2	2	34	245	30	50	.380 Auto	12
3	3	35	788	90	600	7.62 X 54	5
4	4	36	800	60	500	7.62 X 39	10
5	5	37	448	80	30	Buckshot	2

```
--weather
INSERT INTO WEATHER
VALUES (seq_weather.nextval, 'Sunny', 'Clear skies with abundant
sunshine.');
INSERT INTO WEATHER
VALUES (seq_weather.nextval, 'Cloudy', 'Overcast sky with no
direct sunlight.');
INSERT INTO WEATHER
VALUES (seq_weather.nextval, 'Rainy', 'Continuous precipitation
with wet conditions.');
INSERT INTO WEATHER
VALUES (seq_weather.nextval, 'Stormy', 'Violent atmospheric
disturbance: thunder, lightning, and heavy rain.');
INSERT INTO WEATHER
VALUES (seq_weather.nextval, 'Foggy', 'Thick fog reducing
visibility.');
```



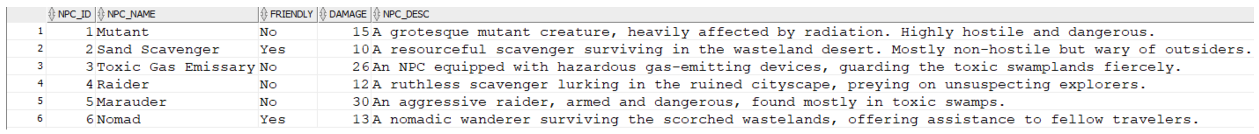
	WEATHER_ID	WEATHER_NAME	WEATHER_DESC
1	1	Sunny	Clear skies with abundant sunshine.
2	2	Cloudy	Overcast sky with no direct sunlight.
3	3	Rainy	Continuous precipitation with wet conditions.
4	4	Stormy	Violent atmospheric disturbance: thunder, lightning, and heavy rain.
5	5	Foggy	Thick fog reducing visibility.

```
--biomes
INSERT INTO BIOME
VALUES (seq_biome.nextval, 'Forest', 'A twisted and eerie
forest, overrun with dangerous creatures.');
INSERT INTO BIOME
VALUES (seq_biome.nextval, 'Wasteland Desert', 'A desolate and
barren wasteland, scattered with remnants of civilization and
plagued by sandstorms.');
INSERT INTO BIOME
VALUES (seq_biome.nextval, 'Toxic Swamplands', 'A toxic and
hazardous swamp, filled with poisonous gases, mutated creatures,
and decaying ruins.');
INSERT INTO BIOME
VALUES (seq_biome.nextval, 'Ruined Cityscape', 'The remnants of
a once thriving city, now reduced to ruins, rubble, and danger
at every turn.');
INSERT INTO BIOME
VALUES (seq_biome.nextval, 'Industrial Zone', 'An industrial
area in decay, filled with waste, malfunctioning machinery, and
hostile scavengers.');
```



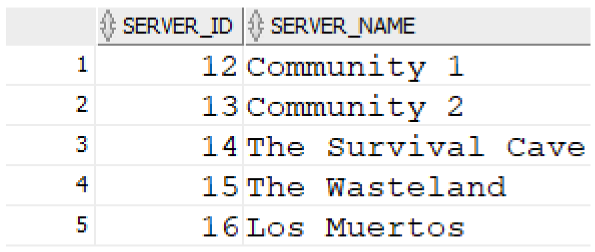
	BIOME_ID	BIOME_NAME	BIOME_DESC
1	1	Forest	A twisted and eerie forest, overrun with dangerous creatures.
2	2	Wasteland Desert	A desolate and barren wasteland, scattered with remnants of civilization and plagued by sandstorms.
3	3	Toxic Swamplands	A toxic and hazardous swamp, filled with poisonous gases, mutated creatures, and decaying ruins.
4	4	Ruined Cityscape	The remnants of a once thriving city, now reduced to ruins, rubble, and danger at every turn.
5	5	Industrial Zone	An industrial area in decay, filled with waste, malfunctioning machinery, and hostile scavengers.

```
--npc
INSERT INTO NPC
VALUES (seq_npc.nextval, 'Mutant', 'No', 15, 'A grotesque mutant
creature, heavily affected by radiation. Highly hostile and
dangerous.');
INSERT INTO NPC
VALUES (seq_npc.nextval, 'Sand Scavenger', 'Yes', 10, 'A
resourceful scavenger surviving in the wasteland desert. Mostly
non-hostile but wary of outsiders.');
INSERT INTO NPC
VALUES (seq_npc.nextval, 'Toxic Gas Emissary', 'No', 26, 'An NPC
equipped with hazardous gas-emitting devices, guarding the toxic
swamplands fiercely.');
INSERT INTO NPC
VALUES (seq_npc.nextval, 'Raider', 'No', 12, 'A ruthless
scavenger lurking in the ruined cityscape, preying on
unsuspecting explorers.');
INSERT INTO NPC
VALUES (seq_npc.nextval, 'Marauder', 'No', 30, 'An aggressive
raider, armed and dangerous, found mostly in toxic swamps.');
INSERT INTO NPC
VALUES (seq_npc.nextval, 'Nomad', 'Yes', 13, 'A nomadic wanderer
surviving the scorched wastelands, offering assistance to fellow
travelers.');
```



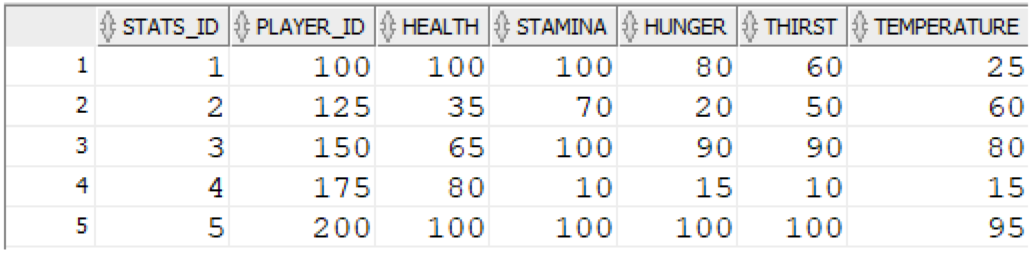
	NPC_ID	NPC_NAME	FRIENDLY	DAMAGE	NPC_DESC
1	1	Mutant	No	15	A grotesque mutant creature, heavily affected by radiation. Highly hostile and dangerous.
2	2	Sand Scavenger	Yes	10	A resourceful scavenger surviving in the wasteland desert. Mostly non-hostile but wary of outsiders.
3	3	Toxic Gas Emissary	No	26	An NPC equipped with hazardous gas-emitting devices, guarding the toxic swamplands fiercely.
4	4	Raider	No	12	A ruthless scavenger lurking in the ruined cityscape, preying on unsuspecting explorers.
5	5	Marauder	No	30	An aggressive raider, armed and dangerous, found mostly in toxic swamps.
6	6	Nomad	Yes	13	A nomadic wanderer surviving the scorched wastelands, offering assistance to fellow travelers.

```
--server
ALTER TABLE SERVER DROP COLUMN Player_ID;
ALTER TABLE SERVER DROP COLUMN Log_Time;
ALTER TABLE SERVER ADD Server_Name varchar2(20);
INSERT INTO SERVER VALUES(seq_server.nextval, 'Community 1');
INSERT INTO SERVER VALUES(seq_server.nextval, 'Community 2');
INSERT INTO SERVER VALUES(seq_server.nextval, 'The Survival
Cave');
INSERT INTO SERVER VALUES(seq_server.nextval, 'The Wasteland');
INSERT INTO SERVER VALUES(seq_server.nextval, 'Los Muertos');
```



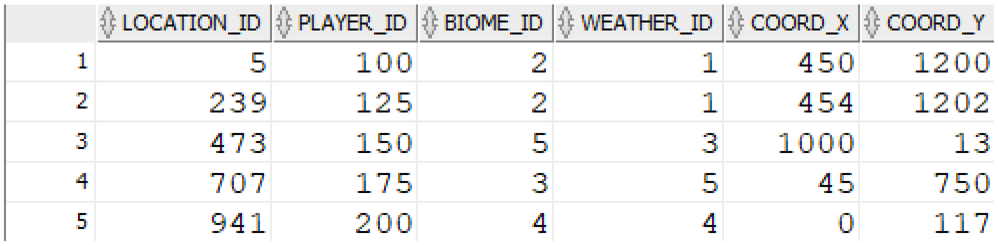
	SERVER_ID	SERVER_NAME
1	12	Community 1
2	13	Community 2
3	14	The Survival Cave
4	15	The Wasteland
5	16	Los Muertos

```
--stats
INSERT INTO STATS VALUES(seq_stats.nextval, 100, 100, 100, 80,
60, 25);
INSERT INTO STATS VALUES(seq_stats.nextval, 125, 35, 70, 20, 50,
60);
INSERT INTO STATS VALUES(seq_stats.nextval, 150, 65, 100, 90,
90, 80);
INSERT INTO STATS VALUES(seq_stats.nextval, 175, 80, 10, 15, 10,
15);
INSERT INTO STATS VALUES(seq_stats.nextval, 200, 100, 100, 100,
100, 95);
```



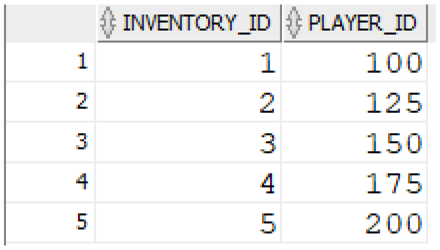
	STATS_ID	PLAYER_ID	HEALTH	STAMINA	HUNGER	THIRST	TEMPERATURE
1	1	100	100	100	80	60	25
2	2	125	35	70	20	50	60
3	3	150	65	100	90	90	80
4	4	175	80	10	15	10	15
5	5	200	100	100	100	100	95

```
--location
INSERT INTO LOCATION VALUES(seq_location.nextval, 100, 2, 1,
450, 1200);
INSERT INTO LOCATION VALUES(seq_location.nextval, 125, 2, 1,
454, 1202);
INSERT INTO LOCATION VALUES(seq_location.nextval, 150, 5, 3,
1000, 13);
INSERT INTO LOCATION VALUES(seq_location.nextval, 175, 3, 5, 45,
750);
INSERT INTO LOCATION VALUES(seq_location.nextval, 200, 4, 4, 0,
117);
```



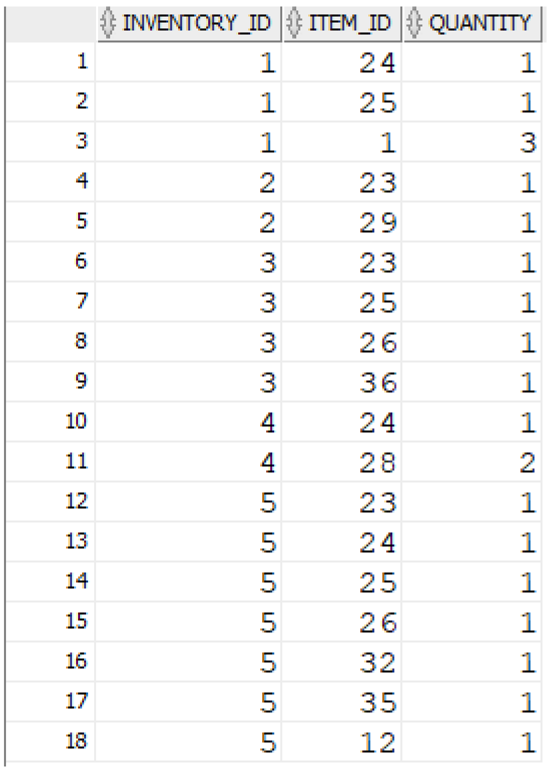
	LOCATION_ID	PLAYER_ID	BIOME_ID	WEATHER_ID	COORD_X	COORD_Y
1	5	100	2	1	450	1200
2	239	125	2	1	454	1202
3	473	150	5	3	1000	13
4	707	175	3	5	45	750
5	941	200	4	4	0	117

```
--inventory
INSERT INTO INVENTORY VALUES(seq_inventory.nextval, 100);
INSERT INTO INVENTORY VALUES(seq_inventory.nextval, 125);
INSERT INTO INVENTORY VALUES(seq_inventory.nextval, 150);
INSERT INTO INVENTORY VALUES(seq_inventory.nextval, 175);
INSERT INTO INVENTORY VALUES(seq_inventory.nextval, 200);
```



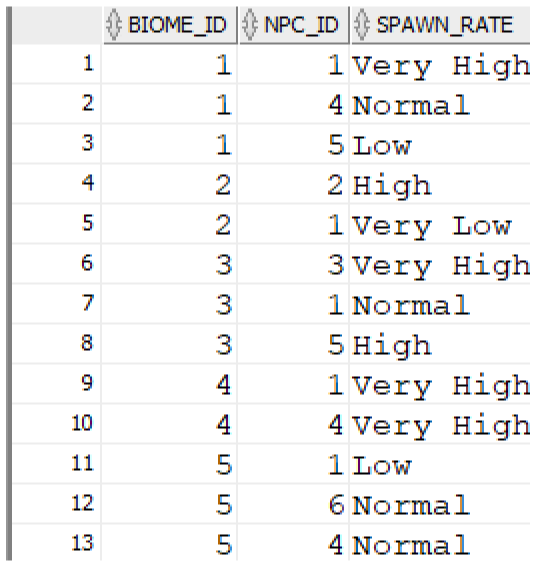
	INVENTORY_ID	PLAYER_ID
1	1	100
2	2	125
3	3	150
4	4	175
5	5	200

```
--contains
INSERT INTO CONTAINS VALUES(1, 24, 1);
INSERT INTO CONTAINS VALUES(1, 25, 1);
INSERT INTO CONTAINS VALUES(1, 1, 3);
INSERT INTO CONTAINS VALUES(2, 23, 1);
INSERT INTO CONTAINS VALUES(2, 29, 1);
INSERT INTO CONTAINS VALUES(3, 23, 1);
INSERT INTO CONTAINS VALUES(3, 25, 1);
INSERT INTO CONTAINS VALUES(3, 26, 1);
INSERT INTO CONTAINS VALUES(3, 36, 1);
INSERT INTO CONTAINS VALUES(4, 24, 1);
INSERT INTO CONTAINS VALUES(4, 28, 2);
INSERT INTO CONTAINS VALUES(5, 23, 1);
INSERT INTO CONTAINS VALUES(5, 24, 1);
INSERT INTO CONTAINS VALUES(5, 25, 1);
INSERT INTO CONTAINS VALUES(5, 26, 1);
INSERT INTO CONTAINS VALUES(5, 32, 1);
INSERT INTO CONTAINS VALUES(5, 35, 1);
INSERT INTO CONTAINS VALUES(5, 12, 1);
```



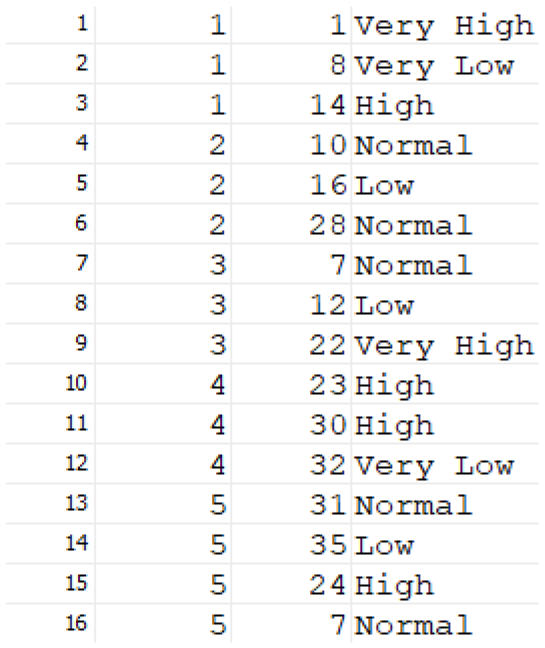
	INVENTORY_ID	ITEM_ID	QUANTITY
1	1	24	1
2	1	25	1
3	1	1	3
4	2	23	1
5	2	29	1
6	3	23	1
7	3	25	1
8	3	26	1
9	3	36	1
10	4	24	1
11	4	28	2
12	5	23	1
13	5	24	1
14	5	25	1
15	5	26	1
16	5	32	1
17	5	35	1
18	5	12	1

```
--npc_in_biomes
INSERT INTO NPC_IN_BIOMES VALUES(1, 1, 'Very High');
INSERT INTO NPC_IN_BIOMES VALUES(1, 4, 'Normal');
INSERT INTO NPC_IN_BIOMES VALUES(1, 5, 'Low');
INSERT INTO NPC_IN_BIOMES VALUES(2, 2, 'High');
INSERT INTO NPC_IN_BIOMES VALUES(2, 1, 'Very Low');
INSERT INTO NPC_IN_BIOMES VALUES(3, 3, 'Very High');
INSERT INTO NPC_IN_BIOMES VALUES(3, 1, 'Normal');
INSERT INTO NPC_IN_BIOMES VALUES(3, 5, 'High');
INSERT INTO NPC_IN_BIOMES VALUES(4, 1, 'Very High');
INSERT INTO NPC_IN_BIOMES VALUES(4, 4, 'Very High');
INSERT INTO NPC_IN_BIOMES VALUES(5, 1, 'Low');
INSERT INTO NPC_IN_BIOMES VALUES(5, 6, 'Normal');
INSERT INTO NPC_IN_BIOMES VALUES(5, 4, 'Normal');
```



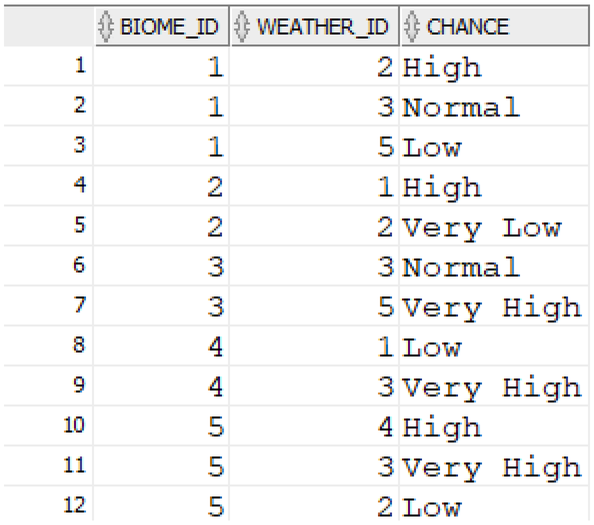
	BIOME_ID	NPC_ID	SPAWN_RATE
1	1	1	Very High
2	1	4	Normal
3	1	5	Low
4	2	2	High
5	2	1	Very Low
6	3	3	Very High
7	3	1	Normal
8	3	5	High
9	4	1	Very High
10	4	4	Very High
11	5	1	Low
12	5	6	Normal
13	5	4	Normal

```
--items-in-biomes
INSERT INTO ITEMS_IN_BIOMES VALUES(1, 1, 'Very High');
INSERT INTO ITEMS_IN_BIOMES VALUES(1, 8, 'Very Low');
INSERT INTO ITEMS_IN_BIOMES VALUES(1, 14, 'High');
INSERT INTO ITEMS_IN_BIOMES VALUES(2, 10, 'Normal');
INSERT INTO ITEMS_IN_BIOMES VALUES(2, 16, 'Low');
INSERT INTO ITEMS_IN_BIOMES VALUES(2, 28, 'Normal');
INSERT INTO ITEMS_IN_BIOMES VALUES(3, 7, 'Normal');
INSERT INTO ITEMS_IN_BIOMES VALUES(3, 12, 'Low');
INSERT INTO ITEMS_IN_BIOMES VALUES(3, 22, 'Very High');
INSERT INTO ITEMS_IN_BIOMES VALUES(4, 23, 'High');
INSERT INTO ITEMS_IN_BIOMES VALUES(4, 30, 'High');
INSERT INTO ITEMS_IN_BIOMES VALUES(4, 32, 'Very Low');
INSERT INTO ITEMS_IN_BIOMES VALUES(5, 31, 'Normal');
INSERT INTO ITEMS_IN_BIOMES VALUES(5, 35, 'Low');
INSERT INTO ITEMS_IN_BIOMES VALUES(5, 24, 'High');
INSERT INTO ITEMS_IN_BIOMES VALUES(5, 7, 'Normal');
```



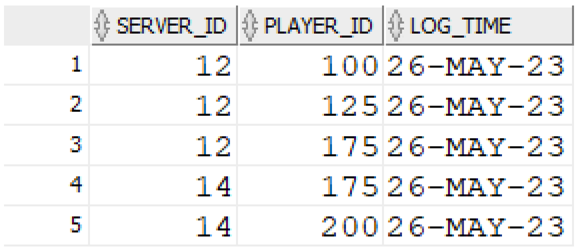
1	1	1	Very High
2	1	8	Very Low
3	1	14	High
4	2	10	Normal
5	2	16	Low
6	2	28	Normal
7	3	7	Normal
8	3	12	Low
9	3	22	Very High
10	4	23	High
11	4	30	High
12	4	32	Very Low
13	5	31	Normal
14	5	35	Low
15	5	24	High
16	5	7	Normal

```
--biomes_weather
INSERT INTO BIOMES_WEATHER VALUES(1, 2, 'High');
INSERT INTO BIOMES_WEATHER VALUES(1, 3, 'Normal');
INSERT INTO BIOMES_WEATHER VALUES(1, 5, 'Low');
INSERT INTO BIOMES_WEATHER VALUES(2, 1, 'High');
INSERT INTO BIOMES_WEATHER VALUES(2, 2, 'Very Low');
INSERT INTO BIOMES_WEATHER VALUES(3, 3, 'Normal');
INSERT INTO BIOMES_WEATHER VALUES(3, 5, 'Very High');
INSERT INTO BIOMES_WEATHER VALUES(4, 1, 'Low');
INSERT INTO BIOMES_WEATHER VALUES(4, 3, 'Very High');
INSERT INTO BIOMES_WEATHER VALUES(5, 4, 'High');
INSERT INTO BIOMES_WEATHER VALUES(5, 3, 'Very High');
INSERT INTO BIOMES_WEATHER VALUES(5, 2, 'Low');
```



	BIOME_ID	WEATHER_ID	CHANCE
1	1	2	High
2	1	3	Normal
3	1	5	Low
4	2	1	High
5	2	2	Very Low
6	3	3	Normal
7	3	5	Very High
8	4	1	Low
9	4	3	Very High
10	5	4	High
11	5	3	Very High
12	5	2	Low

```
--server_connections
INSERT INTO SERVER_CONNECTIONS VALUES(12, 100,
TO_DATE('2023-05-26 08:15:00', 'YYYY-MM-DD HH24:MI:SS'));
INSERT INTO SERVER_CONNECTIONS VALUES(12, 125,
TO_DATE('2023-05-26 09:20:30', 'YYYY-MM-DD HH24:MI:SS'));
INSERT INTO SERVER_CONNECTIONS VALUES(12, 175,
TO_DATE('2023-05-26 10:10:32', 'YYYY-MM-DD HH24:MI:SS'));
INSERT INTO SERVER_CONNECTIONS VALUES(14, 175,
TO_DATE('2023-05-26 13:40:37', 'YYYY-MM-DD HH24:MI:SS'));
INSERT INTO SERVER_CONNECTIONS VALUES(14, 200,
TO_DATE('2023-05-26 15:55:00', 'YYYY-MM-DD HH24:MI:SS'));

Commit;
```



	SERVER_ID	PLAYER_ID	LOG_TIME
1	12	100	26-MAY-23
2	12	125	26-MAY-23
3	12	175	26-MAY-23
4	14	175	26-MAY-23
5	14	200	26-MAY-23

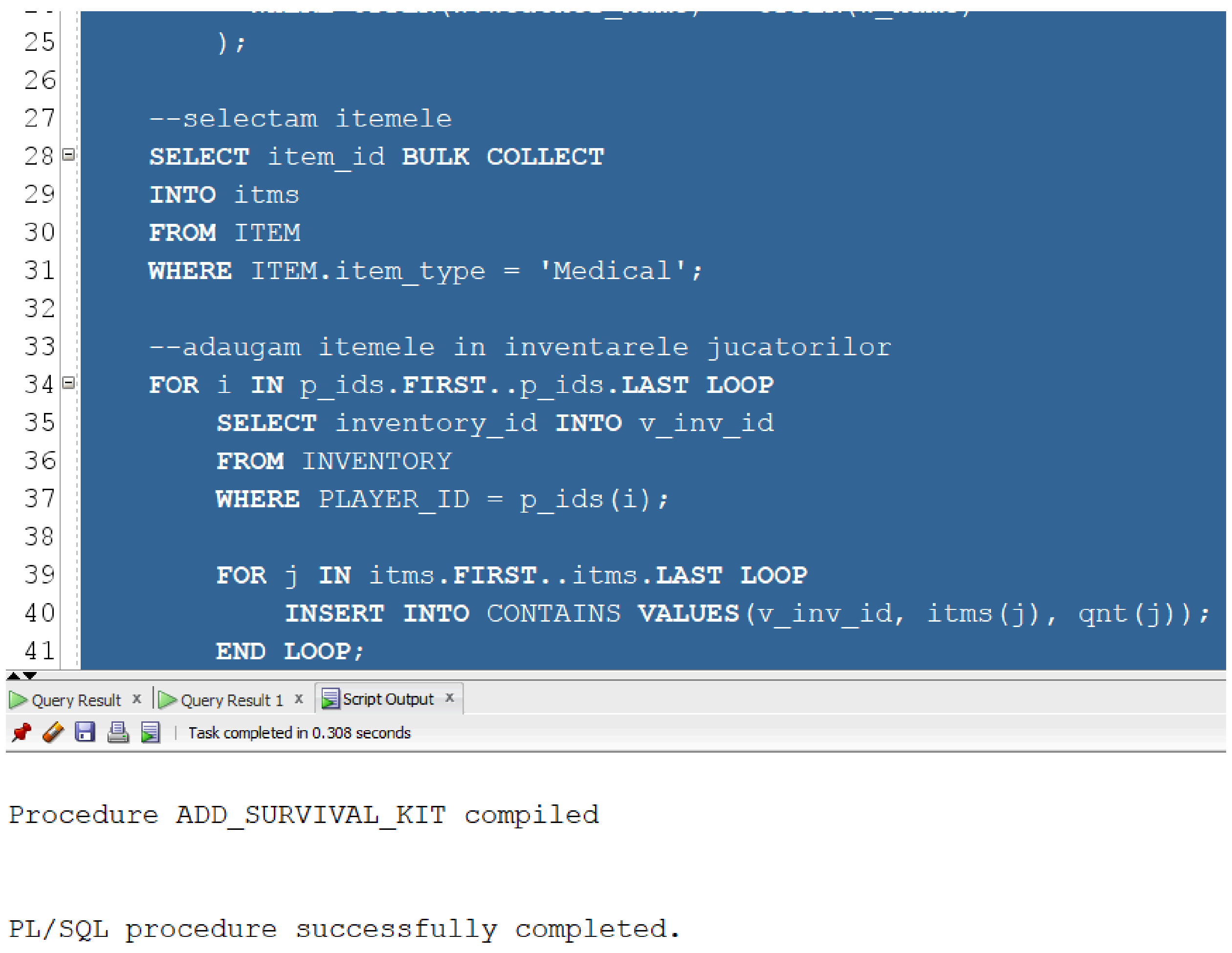
6. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate. Apelați subprogramul.

Scrieți o procedură care primește un tip de vreme (Sunny, Cloudy, Rainy, Stormy, Foggy) și adaugă în inventarul fiecărui jucător aflat într-o locație cu acel tip de vreme un survival kit care conține fiecare item medical din joc cel puțin o dată.

```
CREATE OR REPLACE PROCEDURE add_survival_kit
    (w_name WEATHER.weather_name%TYPE)
    IS
        --tablou indexat pentru a retine id-urile jucatorilor
        TYPE player_ids IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
        --tablou imbricat care retine itemele din kit
        TYPE kit_items IS TABLE OF NUMBER;
        --vector pentru a retine cate iteme de fiecare tip
        --sunt adaugate
        TYPE items_vect IS VARRAY(10) OF NUMBER;

        CURSOR med_items(pid PLAYER.player_id%TYPE) IS
            SELECT it.item_id
            FROM ITEM it
            JOIN CONTAINS c on c.item_id = it.item_id
            JOIN INVENTORY i on i.inventory_id = c.inventory_id
            WHERE i.player_id = pid AND it.item_type =
'Medical';

        qnt         items_vect := items_vect(1, 1, 2, 1, 1);
        itms        kit_items;
        p_ids       player_ids;
        v_inv_id    CONTAINS.inventory_id%TYPE;
        v_exists    BOOLEAN;
    BEGIN
        --selectam jucatorii care se afla in locatii
        --cu vremea primita ca argument
        SELECT PLAYER_ID BULK COLLECT
        INTO p_ids
        FROM (SELECT L.PLAYER_ID
              FROM LOCATION L
              JOIN WEATHER W ON W.weather_id = L.weather_id
              WHERE UPPER(W.weather_name) = UPPER(w_name)
            );

        --selectam itemele
        SELECT item_id BULK COLLECT
        INTO itms
        FROM ITEM
        WHERE ITEM.item_type = 'Medical';

        --adaugam itemele in inventarele jucatorilor
        FOR i IN p_ids.FIRST..p_ids.LAST LOOP
            SELECT inventory_id INTO v_inv_id
            FROM INVENTORY
            WHERE PLAYER_ID = p_ids(i);

            FOR j IN itms.FIRST..itms.LAST LOOP
                v_exists := FALSE;

                FOR k IN med_items(p_ids(i)) LOOP
                    IF k.item_id = itms(j) THEN
                        v_exists := TRUE;
                        EXIT;
                    END IF;
                END LOOP;

                --daca exista unul din iteme in inventar
                IF v_exists THEN
                    CONTINUE;
                END IF;

                INSERT INTO CONTAINS VALUES(v_inv_id, itms(j),
qnt(j));
            END LOOP;
        END LOOP;
    END add_survival_kit;
/

BEGIN
    add_survival_kit('Rainy');
END;
```

/

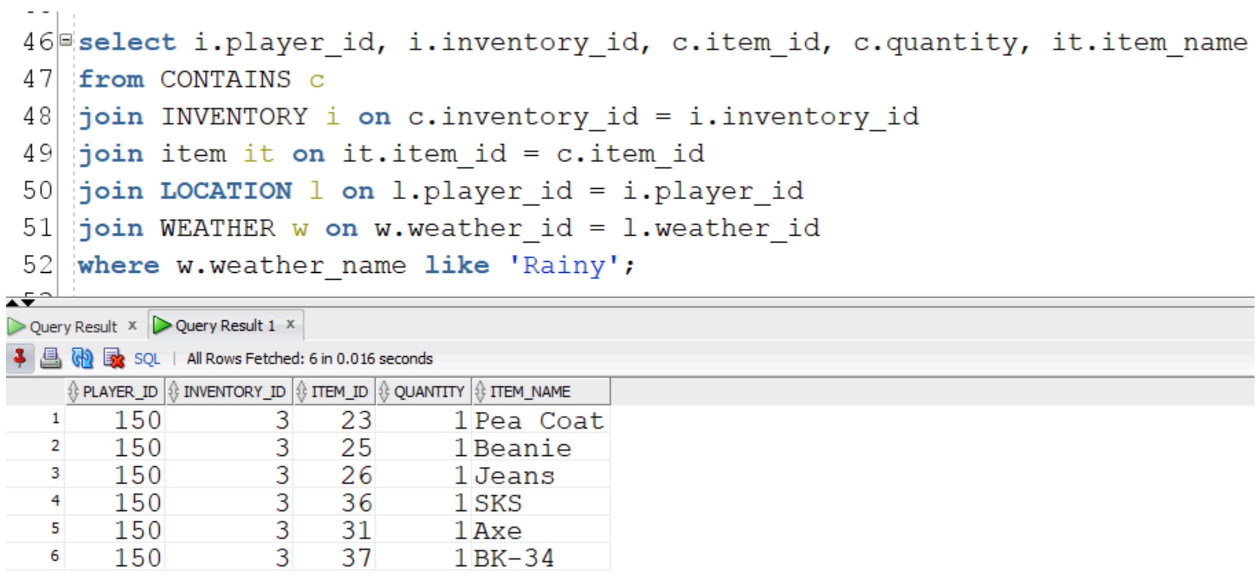
În implementarea aleasă am folosit vectorul de cantități qnt pentru a seta cantitatea fiecărui item medical primit de jucători.



```
        );

    --selectam itemele
    SELECT item_id BULK COLLECT
    INTO itms
    FROM ITEM
    WHERE ITEM.item_type = 'Medical';

    --adaugam itemele in inventarele jucatorilor
    FOR i IN p_ids.FIRST..p_ids.LAST LOOP
        SELECT inventory_id INTO v_inv_id
        FROM INVENTORY
        WHERE PLAYER_ID = p_ids(i);

        FOR j IN itms.FIRST..itms.LAST LOOP
            INSERT INTO CONTAINS VALUES(v_inv_id, itms(j), qnt(j));
        END LOOP;
```

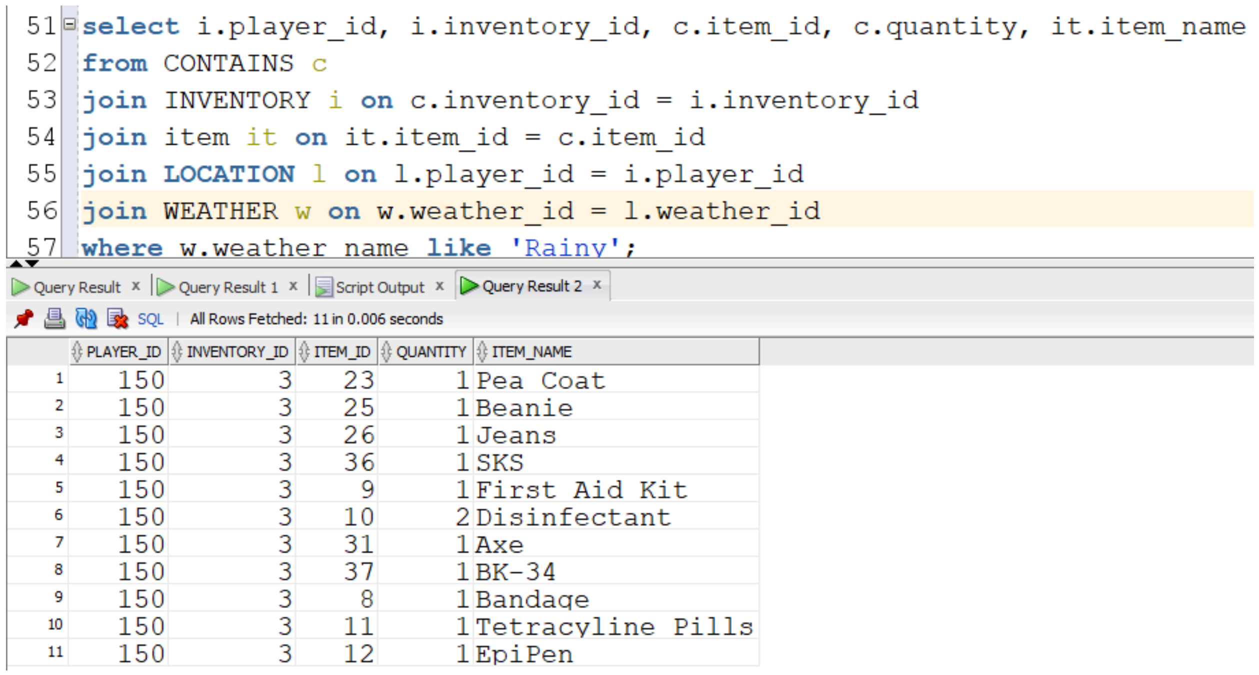
Task completed in 0.308 seconds

```
Procedure ADD_SURVIVAL_KIT compiled


PL/SQL procedure successfully completed.
```


```sql
select i.player_id, i.inventory_id, c.item_id, c.quantity, it.item_name
from CONTAINS c
join INVENTORY i on c.inventory_id = i.inventory_id
join item it on it.item_id = c.item_id
join LOCATION l on l.player_id = i.player_id
join WEATHER w on w.weather_id = l.weather_id
where w.weather_name like 'Rainy';
```

All Rows Fetched: 6 in 0.016 seconds

	PLAYER_ID	INVENTORY_ID	ITEM_ID	QUANTITY	ITEM_NAME
1	150	3	23	1	Pea Coat
2	150	3	25	1	Beanie
3	150	3	26	1	Jeans
4	150	3	36	1	SKS
5	150	3	31	1	Axe
6	150	3	37	1	BK-34


```sql
select i.player_id, i.inventory_id, c.item_id, c.quantity, it.item_name
from CONTAINS c
join INVENTORY i on c.inventory_id = i.inventory_id
join item it on it.item_id = c.item_id
join LOCATION l on l.player_id = i.player_id
join WEATHER w on w.weather_id = l.weather_id
where w.weather_name like 'Rainy';
```

All Rows Fetched: 11 in 0.006 seconds

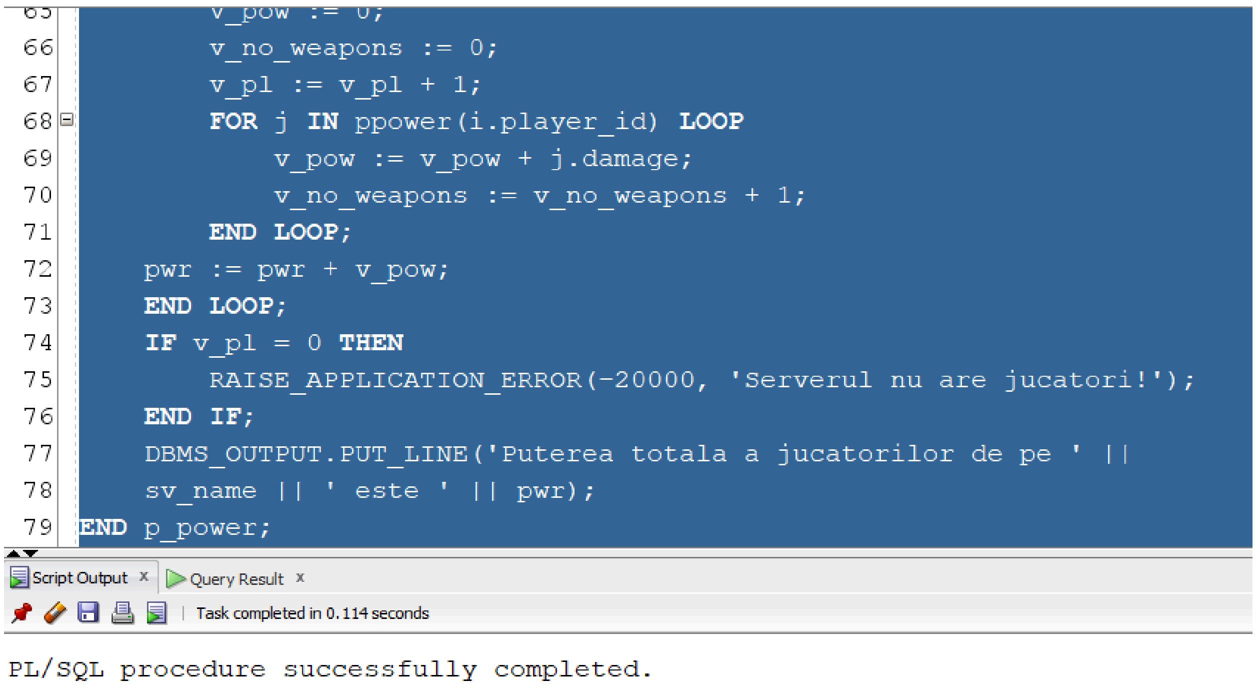
	PLAYER_ID	INVENTORY_ID	ITEM_ID	QUANTITY	ITEM_NAME
1	150	3	23	1	Pea Coat
2	150	3	25	1	Beanie
3	150	3	26	1	Jeans
4	150	3	36	1	SKS
5	150	3	9	1	First Aid Kit
6	150	3	10	2	Disinfectant
7	150	3	31	1	Axe
8	150	3	37	1	BK-34
9	150	3	8	1	Bandage
10	150	3	11	1	Tetracyline Pills
11	150	3	12	1	EpiPen

7. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor. Apelați subprogramul.

Scrieți o procedură care primește ca parametru numele unui server și afișează puterea totală a jucătorilor de pe acel server.

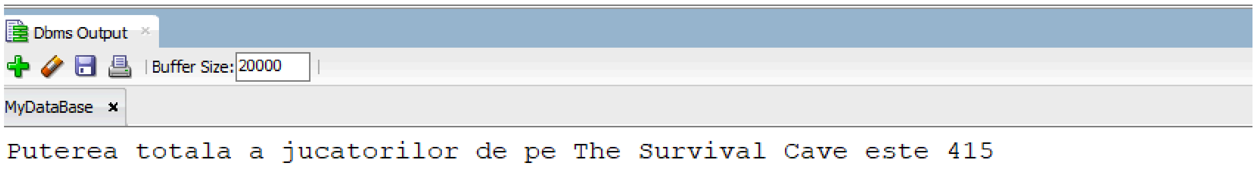
```
CREATE OR REPLACE PROCEDURE p_power
    (sv_name  SERVER.server_name%TYPE)
IS
    pwr        FIREARMS.damage%TYPE;

    CURSOR ppower(pid PLAYER.player_id%TYPE) IS
        SELECT damage
        FROM FIREARMS f
        JOIN ITEM it ON it.item_id = f.item_id
        JOIN CONTAINS c ON c.item_id = it.item_id
        JOIN INVENTORY i ON i.inventory_id = c.inventory_id
        WHERE i.player_id = pid
        UNION
        SELECT damage
        FROM MELEE_WEAPONS mw
        JOIN ITEM it ON it.item_id = mw.item_id
        JOIN CONTAINS c ON c.item_id = it.item_id
        JOIN INVENTORY i ON i.inventory_id = c.inventory_id
        WHERE i.player_id = pid;

    v_pow               FIREARMS.damage%TYPE;
    v_no_weapons        NUMBER;
    v_pid               PLAYER.player_id%TYPE;
    v_pname             PLAYER.player_name%TYPE;
    v_pl                NUMBER;
BEGIN
    v_pl := 0;
    pwr := 0;
    FOR i IN (  SELECT p.player_name, p.player_id
                FROM PLAYER p
                JOIN SERVER_CONNECTIONS sc ON sc.player_id =
p.player_id
                JOIN SERVER s ON s.server_id = sc.server_id
                WHERE s.server_name = sv_name
```

```
                    )LOOP
        v_pow := 0;
        v_no_weapons := 0;
        v_pl := v_pl + 1;
        FOR j IN ppower(i.player_id) LOOP
            v_pow := v_pow + j.damage;
            v_no_weapons := v_no_weapons + 1;
        END LOOP;
    pwr := pwr + v_pow;
    END LOOP;
    IF v_pl = 0 THEN
        RAISE_APPLICATION_ERROR(-20000, 'Serverul nu are
jucatori!');
    END IF;
    DBMS_OUTPUT.PUT_LINE('Puterea totala a jucatorilor de pe '
||
    sv_name || ' este ' || pwr);
END p_power;
/

EXECUTE p_power('The Survival Cave');
```



```
        v_pow := 0;
        v_no_weapons := 0;
        v_pl := v_pl + 1;
        FOR j IN ppower(i.player_id) LOOP
            v_pow := v_pow + j.damage;
            v_no_weapons := v_no_weapons + 1;
        END LOOP;
    pwr := pwr + v_pow;
    END LOOP;
    IF v_pl = 0 THEN
        RAISE_APPLICATION_ERROR(-20000, 'Serverul nu are jucatori!');
    END IF;
    DBMS_OUTPUT.PUT_LINE('Puterea totala a jucatorilor de pe ' ||
    sv_name || ' este ' || pwr);
END p_power;
```

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



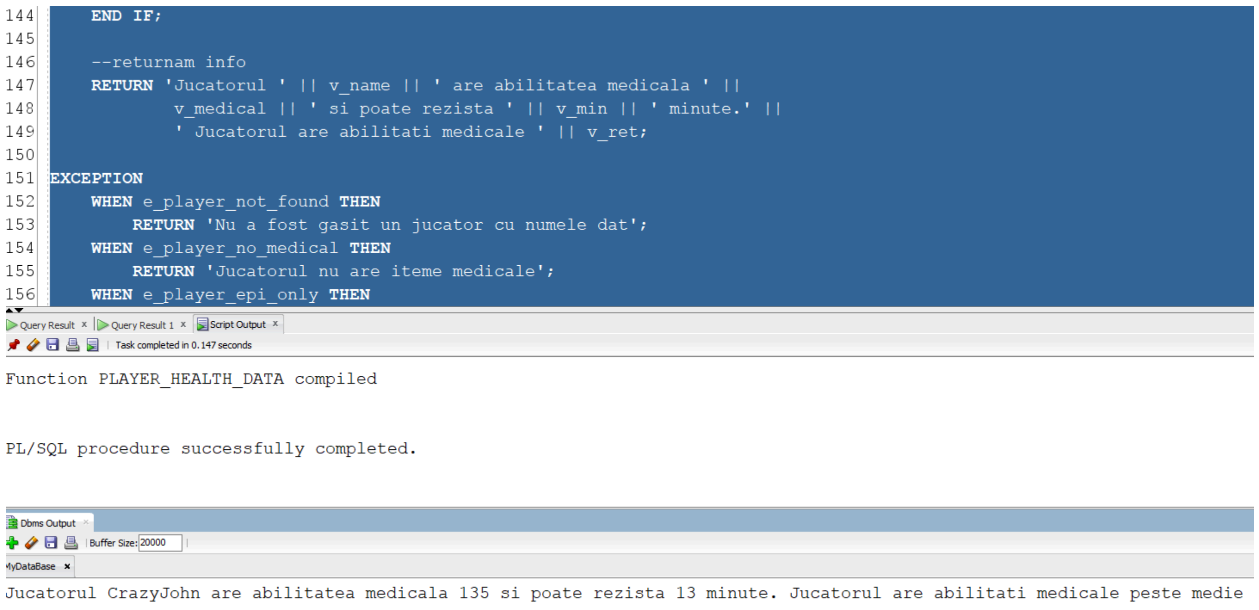
```
Puterea totala a jucatorilor de pe The Survival Cave este 415
```

8. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Definiți minim 2 excepții proprii. Apelați subprogramul astfel încât să evidențiați toate cazurile definite și tratate.

Presupunem că în lumea jocului se declanșează o furtună de gaz toxic care scade viața jucatorilor cu 15 unități la un minut. Scrieți o funcție care ia ca parametru numele unui jucător și afișează nivelul abilităților sale medicale (suma puterii itemelor medicale din inventarul său) și cât timp ar rezista în furtună, în minute. De asemenea, afișați dacă abilitățile medicale ale jucătorului dat sunt sub medie, peste medie, sau medii.

```
CREATE OR REPLACE FUNCTION player_health_data
    (p_name PLAYER.player_name%TYPE)
    RETURN VARCHAR2 IS

    CURSOR player_medical(pid PLAYER.player_id%TYPE) IS
        SELECT m.health_effect
        FROM MEDICAL m
        JOIN ITEM it ON it.item_id = m.item_id
        JOIN CONTAINS c ON c.item_id = it.item_id
        JOIN INVENTORY i ON i.inventory_id = c.inventory_id
        WHERE i.player_id = pid;

    v_medical           MEDICAL.health_effect%TYPE;
    --numarul de iteme medicale din inventar
    v_medical_it        NUMBER;
    v_pid               PLAYER.player_id%TYPE;
    v_name              PLAYER.player_name%TYPE;
    v_health_total      STATS.health%TYPE;
    v_avg_healing       MEDICAL.health_effect%TYPE;
    --numarul de minute supravietuite
    v_min               NUMBER;
    v_ret               VARCHAR2(25);

    --exceptii
    e_player_dead               EXCEPTION;
    e_player_epi_only           EXCEPTION;
    e_player_no_medical         EXCEPTION;
    e_player_not_found          EXCEPTION;

BEGIN
    v_medical := 0;
    v_medical_it := 0;

    --calculam media abilitatilor medicale
    --ale jucatorilor
    SELECt avg(health_ability)
    INTO v_avg_healing
    FROM(
        SELECT p.player_id, sum(m.health_effect) as
health_ability
        FROM player p
        JOIN inventory i on i.player_id = p.player_id
        JOIN contains c on c.inventory_id = i.inventory_id
        JOIN medical m on m.item_id = c.item_id
        GROUP BY p.player_id
        );

    --gasim id-ul si numele jucatorului
    SELECT p.player_id, p.player_name
    INTO v_pid, v_name
    FROM PLAYER p
    WHERE UPPER(p.player_name) LIKE UPPER(p_name);

    --gasim viata initiala a jucatorului
    SELECT s.health
    INTO v_health_total
    FROM STATS s
    WHERE s.player_id = v_pid;

    --calculam abilitatea medicala a jucatorului
    FOR i IN player_medical(v_pid) LOOP
        v_medical := v_medical + i.health_effect;
        v_medical_it := v_medical_it + 1;
    END LOOP;

    IF v_health_total = 0 THEN
        RAISE e_player_dead;
    END IF;

    v_health_total := v_health_total + v_medical;
    v_min := FLOOR(v_health_total / 15);

    IF v_medical < v_avg_healing THEN
        v_ret := 'sub medie';
    ELSIF v_medical = v_avg_healing THEN
        v_ret := 'medii';
    ELSE
        v_ret := 'peste medie';
    END IF;

    IF v_medical = 0 AND v_medical_it = 0 THEN
        RAISE e_player_no_medical;
    ELSIF v_medical = 0 AND v_medical_it <> 0 THEN
        RAISE e_player_epi_only;
    END IF;

    --returnam info
    RETURN 'Jucatorul ' || v_name || ' are abilitatea medicala '
||
            v_medical || ' si poate rezista ' || v_min || '
minute.' ||
            ' Jucatorul are abilitati medicale ' || v_ret;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 'Nu a fost gasit un jucator cu numele dat';
    WHEN e_player_no_medical THEN
        RETURN 'Jucatorul nu are iteme medicale';
    WHEN e_player_epi_only THEN
        RETURN 'Jucatorul are doar EpiPen-uri in inventar';
    WHEN e_player_dead THEN
        RETURN 'Personajul jucatorului nu mai este in viata';
END player_health_data;
/

BEGIN
    DBMS_OUTPUT.PUT_LINE(player_health_data('MagicWizard'));
END;
/
```

Apelul corect:

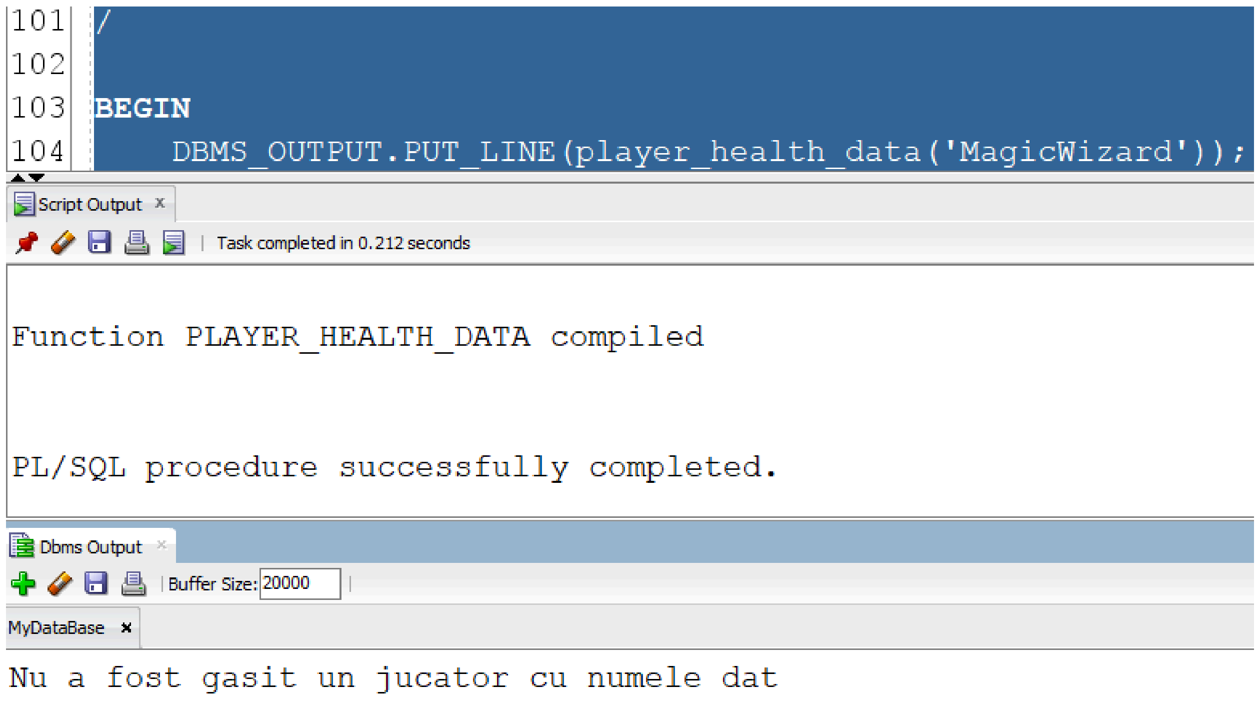


```
    END IF;

    --returnam info
    RETURN 'Jucatorul ' || v_name || ' are abilitatea medicala ' ||
            v_medical || ' si poate rezista ' || v_min || ' minute.' ||
            ' Jucatorul are abilitati medicale ' || v_ret;

EXCEPTION
    WHEN e_player_not_found THEN
        RETURN 'Nu a fost gasit un jucator cu numele dat';
    WHEN e_player_no_medical THEN
        RETURN 'Jucatorul nu are iteme medicale';
    WHEN e_player_epi_only THEN
```

```
Function PLAYER_HEALTH_DATA compiled


PL/SQL procedure successfully completed.
```

```
Jucatorul CrazyJohn are abilitatea medicala 135 si poate rezista 13 minute. Jucatorul are abilitati medicale peste medie
```

Excepția NO_DATA_FOUND

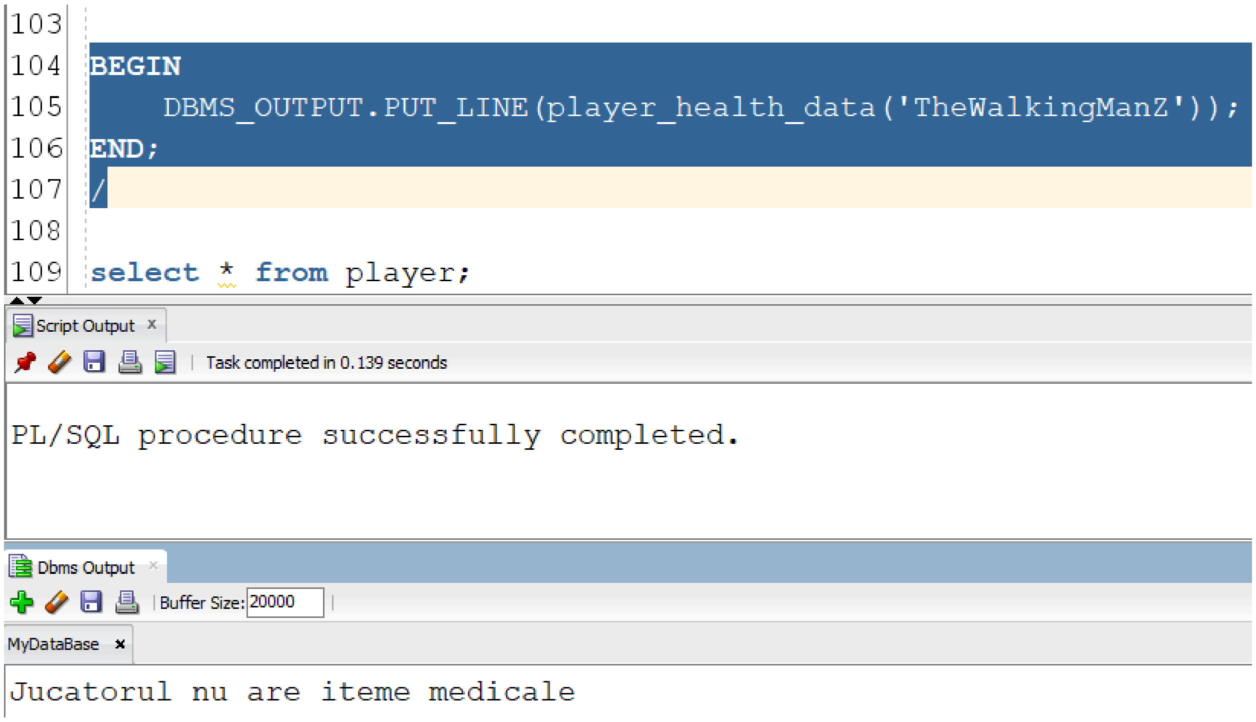


```
/

BEGIN
    DBMS_OUTPUT.PUT_LINE(player_health_data('MagicWizard'));
```

```
Function PLAYER_HEALTH_DATA compiled


PL/SQL procedure successfully completed.
```

```
Nu a fost gasit un jucator cu numele dat
```

Excepția e_player_no_medical



```
BEGIN
    DBMS_OUTPUT.PUT_LINE(player_health_data('TheWalkingManZ'));
END;
/

select * from player;
```

Task completed in 0.139 seconds

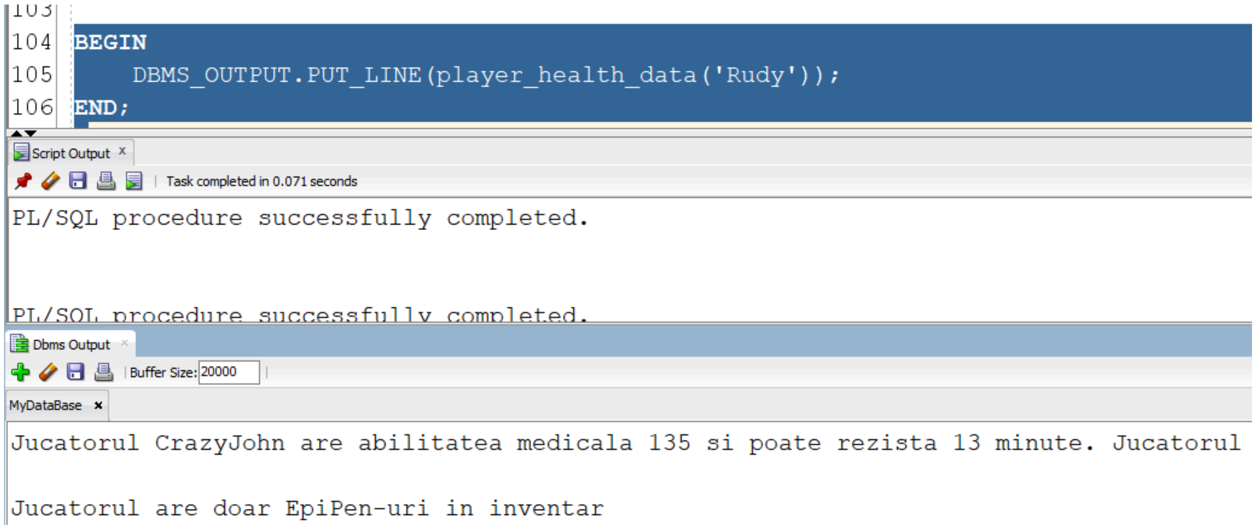
```
PL/SQL procedure successfully completed.
```

Buffer Size: 20000

MyDataBase

```
Jucatorul nu are iteme medicale
```

Excepția e_player_epi_only



```
BEGIN
    DBMS_OUTPUT.PUT_LINE(player_health_data('Rudy'));
END;
```

Task completed in 0.071 seconds

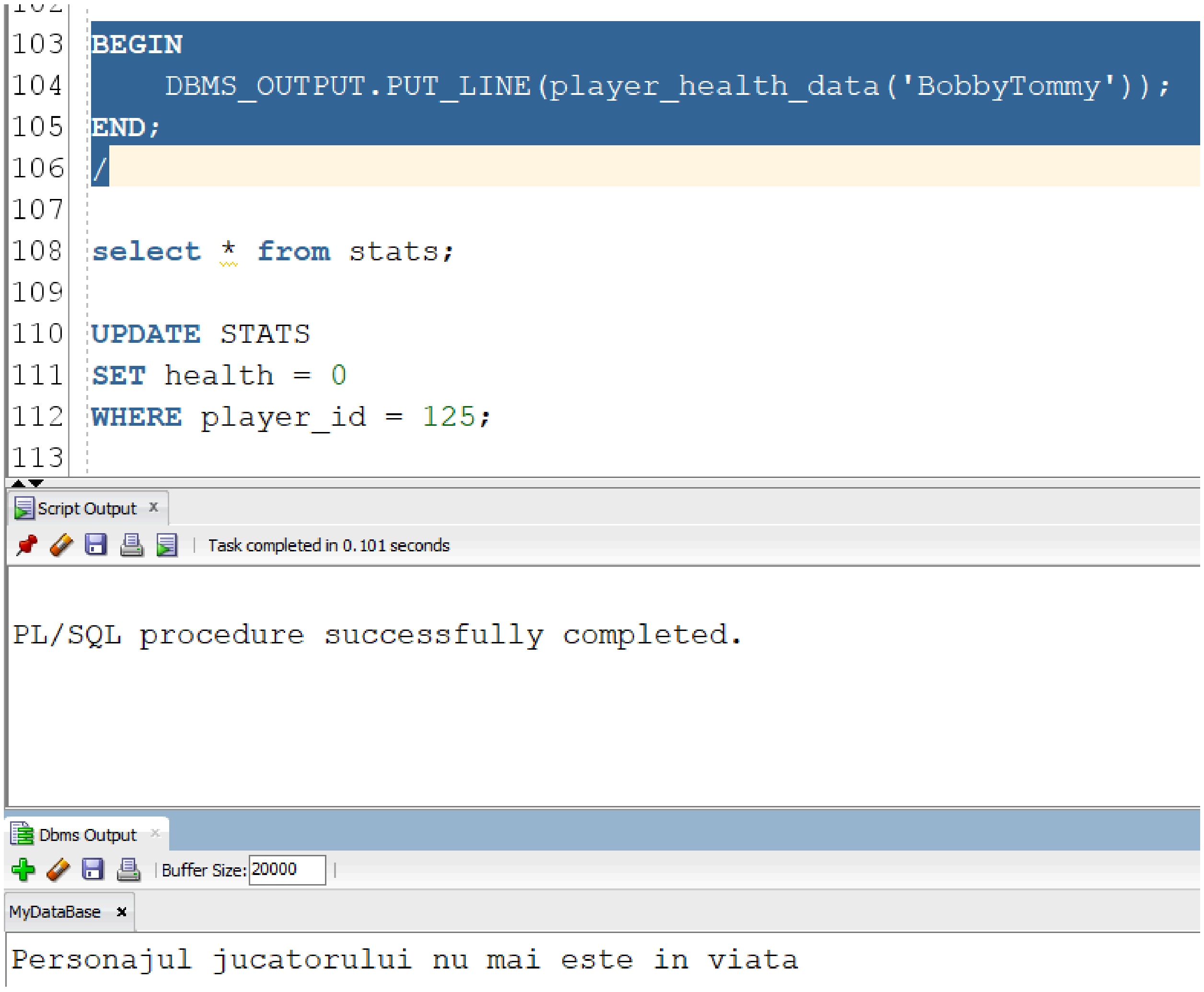
```
PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

Buffer Size: 20000

MyDataBase

```
Jucatorul CrazyJohn are abilitatea medicala 135 si poate rezista 13 minute. Jucatorul

Jucatorul are doar EpiPen-uri in inventar
```

Excepția e_player_dead



```
BEGIN
    DBMS_OUTPUT.PUT_LINE(player_health_data('BobbyTommy'));
END;
/

select * from stats;

UPDATE STATS
SET health = 0
WHERE player_id = 125;
```

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

```
Personajul jucatorului nu mai este in viata
```

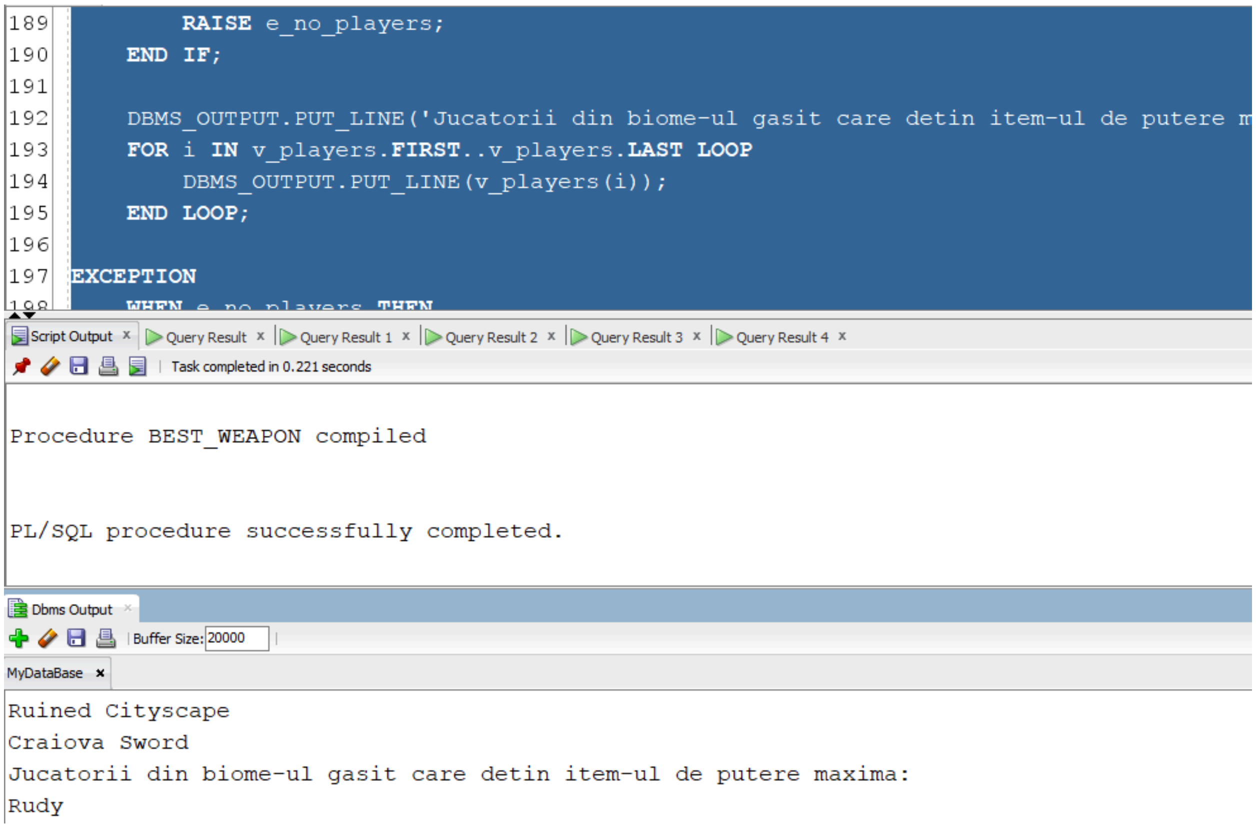
9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea, incluzând excepțiile NO_DATA_FOUND și TOO_MANY_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

Scrieți o procedură care primește un nume de npc și un spawn rate și afișează biome-ul în care se spawnează npc-ul dat cu spawn rate-ul dat, arma cu cel mai mult damage din biome-ul gasit și toți jucătorii care dețin arma respectivă și se află în biome-ul găsit. Folosiți excepții pentru a gestiona cazurile în care există mai multe sau niciun biome și cazurile în care există mai multe sau nicio armă.

```
CREATE OR REPLACE VIEW weapons_damage AS
SELECT f.item_id, it.item_name, f.damage
FROM FIREARMS f
JOIN ITEM it on it.item_id = f.item_id
UNION
SELECT mw.item_id, it.item_name, mw.damage
FROM MELEE_WEAPONS mw
JOIN ITEM it on it.item_id = mw.item_id
ORDER BY damage DESC;

--9
CREATE OR REPLACE PROCEDURE best_weapon
    (n_name NPC.npc_name%TYPE,
    s_rate NPC_IN_BIOMES.spawn_rate%TYPE) IS

    TYPE players_vect IS VARRAY(25) OF VARCHAR2(50);

    v_biome_name        BIOME.biome_name%TYPE;
    v_item_id           ITEM.item_id%TYPE;
    v_item_name         ITEM.item_name%TYPE;
    v_item_dmg          FIREARMS.damage%TYPE;
    v_players           players_vect := players_vect();

    e_no_players        EXCEPTION;

BEGIN
    --gasim biome-ul in care npc-ul dat se spawneaza
    --cu spawn rate-ul dat
    BEGIN
        SELECT b.biome_name
        INTO v_biome_name
        FROM BIOME b
        JOIN NPC_IN_BIOMES nb on nb.biome_id = b.biome_id
        JOIN NPC n on n.npc_id = nb.npc_id
        WHERE UPPER(n.npc_name) LIKE UPPER(n_name)
            AND UPPER(nb.spawn_rate) LIKE UPPER(s_rate);
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('NPC-ul dat nu se spawneaza in
niciun biome cu spawn rate-ul dat.');
            RETURN; --iesim din procedura
        WHEN TOO_MANY_ROWS THEN
            DBMS_OUTPUT.PUT_LINE('Npc-ul dat se spawneaza cu
spawn rate-ul dat in mai multe biome-uri.');
            RETURN;
    END;

    DBMS_OUTPUT.PUT_LINE(v_biome_name);

    --cautam arma cu cel mai mult damage
    --din biome-ul gasit
    BEGIN
        SELECT wd.item_id, wd.item_name
        INTO v_item_id, v_item_name
        FROM weapons_damage wd
        JOIN ITEMS_IN_BIOMES ib on ib.item_id = wd.item_id
        JOIN BIOME b on b.biome_id = ib.biome_id
        WHERE UPPER(b.biome_name) LIKE UPPER(v_biome_name)
        AND wd.damage = (SELECT MAX(wd.damage)
                         FROM weapons_damage wd
                         JOIN ITEMS_IN_BIOMES ib on ib.item_id =
wd.item_id
                         JOIN BIOME b on b.biome_id =
ib.biome_id
                         WHERE UPPER(b.biome_name) LIKE
UPPER(v_biome_name)
                         );
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('Nu se spawneaza arme in
biome-ul gasit.');
            RETURN;
        WHEN TOO_MANY_ROWS THEN
```

```
            DBMS_OUTPUT.PUT_LINE('Se spawneaza mai multe arme cu
putere maxima in biome-ul gasit.');
            RETURN;
    END;

    DBMS_OUTPUT.PUT_LINE(v_item_name);

    --verificam daca arma apare in
    --inventarul unui jucator
    SELECT p.player_name
    BULK COLLECT INTO v_players
    FROM PLAYER p
    JOIN INVENTORY i on i.player_id = p.player_id
    JOIN CONTAINS C on c.inventory_id = i.inventory_id
    JOIN LOCATION l on l.player_id = p.player_id
    JOIN BIOME B on b.biome_id = l.biome_id
    WHERE UPPER(b.biome_name) LIKE UPPER(v_biome_name)
    AND c.item_id = v_item_id;

    IF v_players.COUNT = 0 THEN
        RAISE e_no_players;
    END IF;

    DBMS_OUTPUT.PUT_LINE('Jucatorii din biome-ul gasit care
detin item-ul de putere maxima:');
    FOR i IN v_players.FIRST..v_players.LAST LOOP
        DBMS_OUTPUT.PUT_LINE(v_players(i));
    END LOOP;

EXCEPTION
    WHEN e_no_players THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista jucatori in biome-ul
gasit care sa detine item-ul de putere maxima');
END;
/


EXECUTE best_weapon('Mutant', 'Very High');
```

Apelul fără excepții

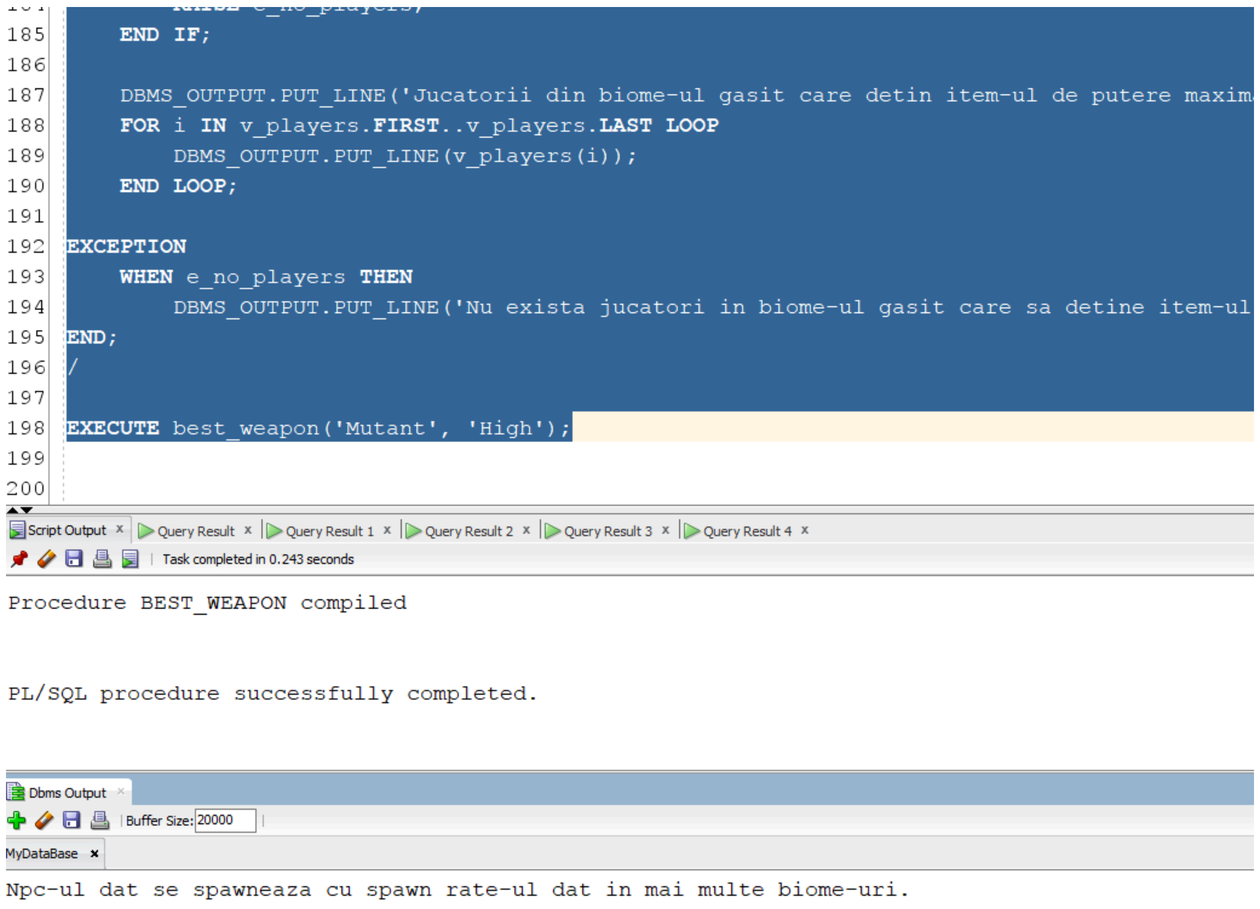


```
        RAISE e_no_players;
    END IF;

    DBMS_OUTPUT.PUT_LINE('Jucatorii din biome-ul gasit care detin item-ul de putere m
    FOR i IN v_players.FIRST..v_players.LAST LOOP
        DBMS_OUTPUT.PUT_LINE(v_players(i));
    END LOOP;

EXCEPTION
    WHEN e_no_players THEN
```

```
Procedure BEST_WEAPON compiled


PL/SQL procedure successfully completed.
```

```
Ruined Cityscape
Craiova Sword
Jucatorii din biome-ul gasit care detin item-ul de putere maxima:
Rudy
```

Excepția TOO_MANY_ROWS la căutarea biome-ului

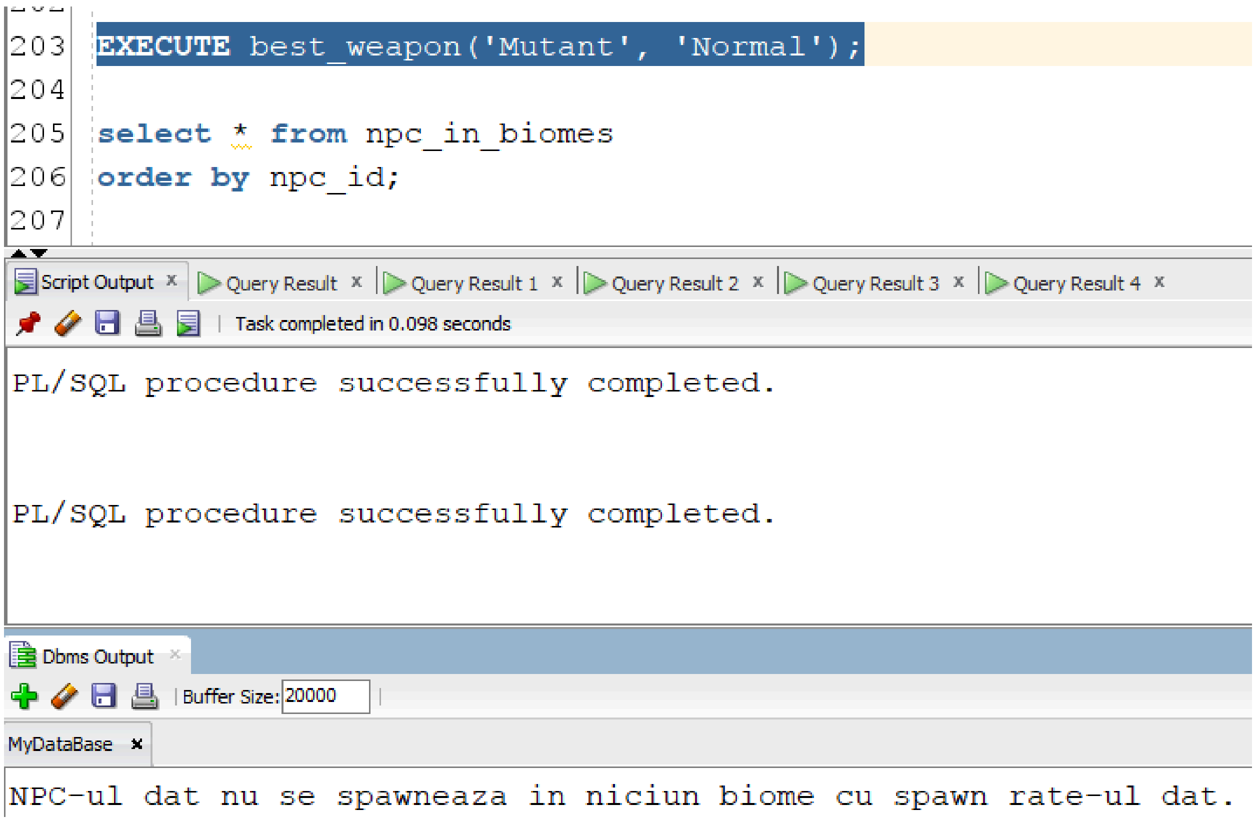


```
    END IF;

    DBMS_OUTPUT.PUT_LINE('Jucatorii din biome-ul gasit care detin item-ul de putere maxim
    FOR i IN v_players.FIRST..v_players.LAST LOOP
        DBMS_OUTPUT.PUT_LINE(v_players(i));
    END LOOP;

EXCEPTION
    WHEN e_no_players THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista jucatori in biome-ul gasit care sa detine item-ul
END;
/

EXECUTE best_weapon('Mutant', 'High');
```

```
Procedure BEST_WEAPON compiled


PL/SQL procedure successfully completed.
```

```
Npc-ul dat se spawneaza cu spawn rate-ul dat in mai multe biome-uri.
```

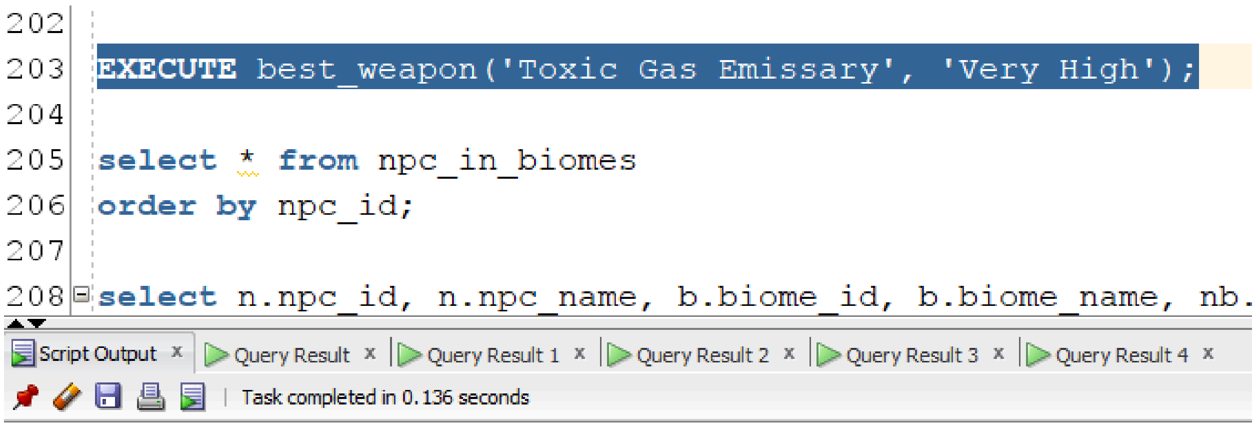
Excepția NO_DATA_FOUND la căutarea biome-ului


```
EXECUTE best_weapon('Mutant', 'Normal');

select * from npc_in_biomes
order by npc_id;
```

```
PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

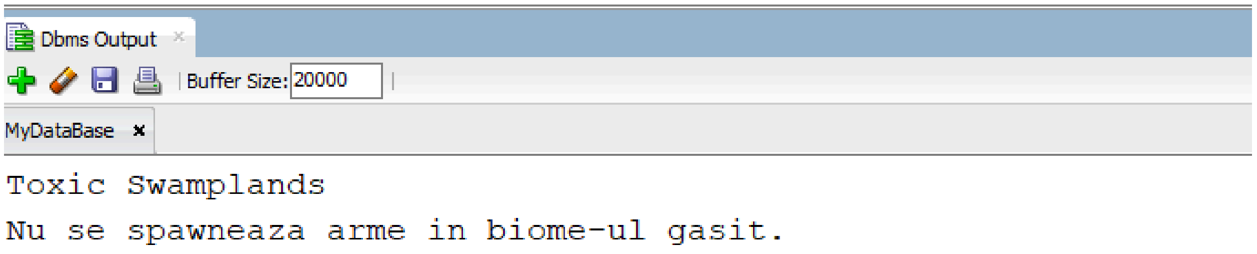
```
NPC-ul dat nu se spawneaza in niciun biome cu spawn rate-ul dat.
```

Excepția NO_DATA_FOUND la cautarea armei



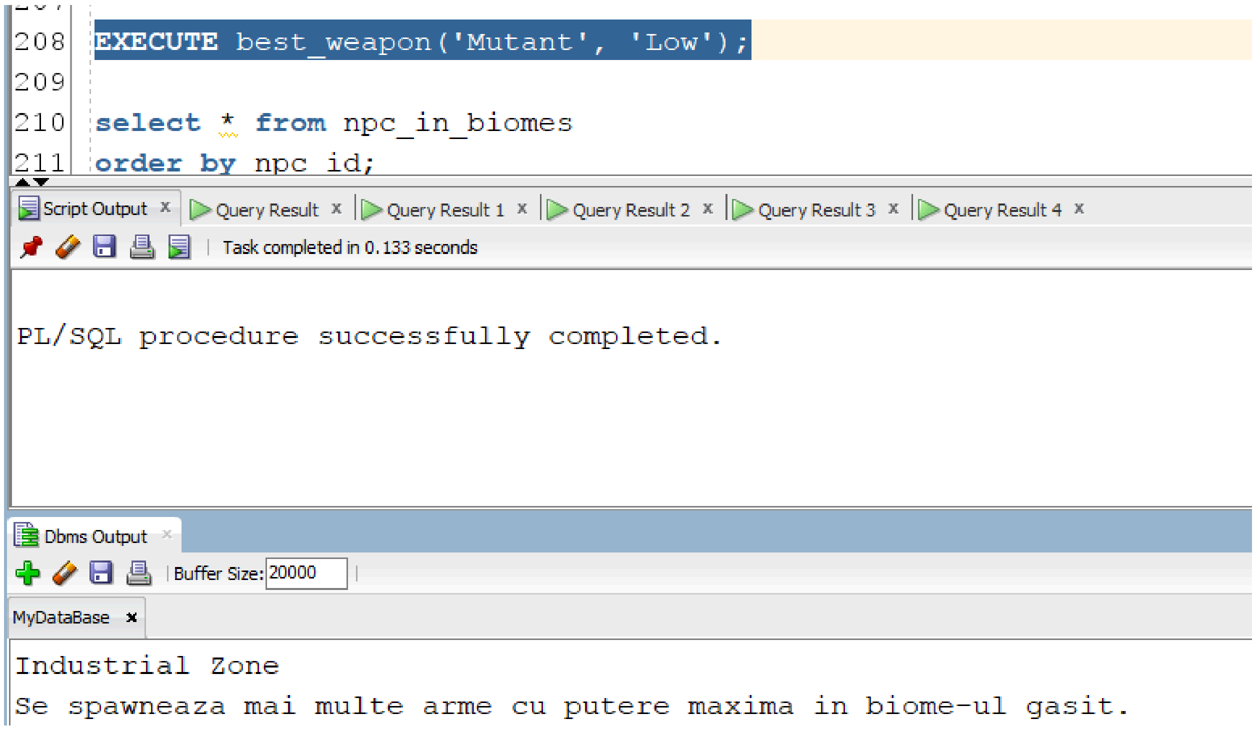
```
EXECUTE best_weapon('Toxic Gas Emissary', 'Very High');

select * from npc_in_biomes
order by npc_id;

select n.npc_id, n.npc_name, b.biome_id, b.biome_name, nb.
```

```
Task completed in 0.136 seconds

PL/SQL procedure successfully completed.
```



```
Toxic Swamplands
Nu se spawneaza arme in biome-ul gasit.
```

Excepția TOO_MANY_ROWS la căutarea armei

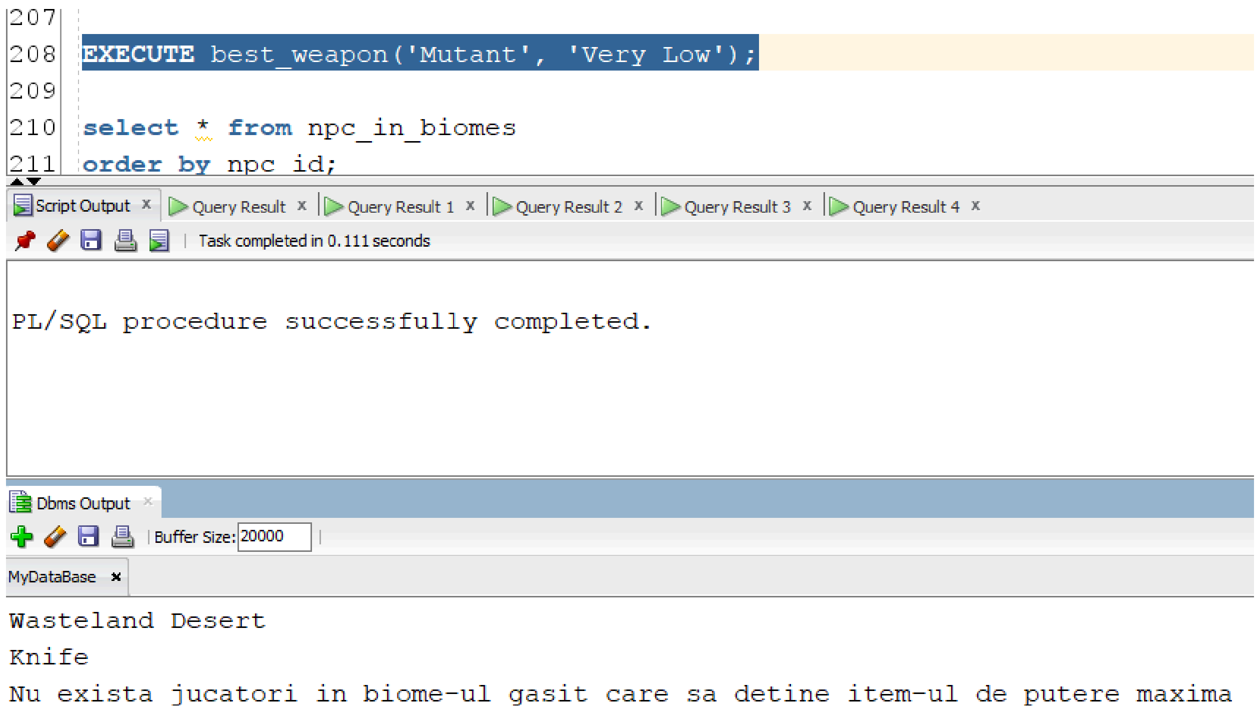


```
EXECUTE best_weapon('Mutant', 'Low');

select * from npc_in_biomes
order by npc_id;
```

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

```
Industrial Zone
Se spawneaza mai multe arme cu putere maxima in biome-ul gasit.
```

Excepția e_no_players - nu există jucători cu arma găsită în biome-ul găsit



```
EXECUTE best_weapon('Mutant', 'Very Low');

select * from npc_in_biomes
order by npc_id;
```

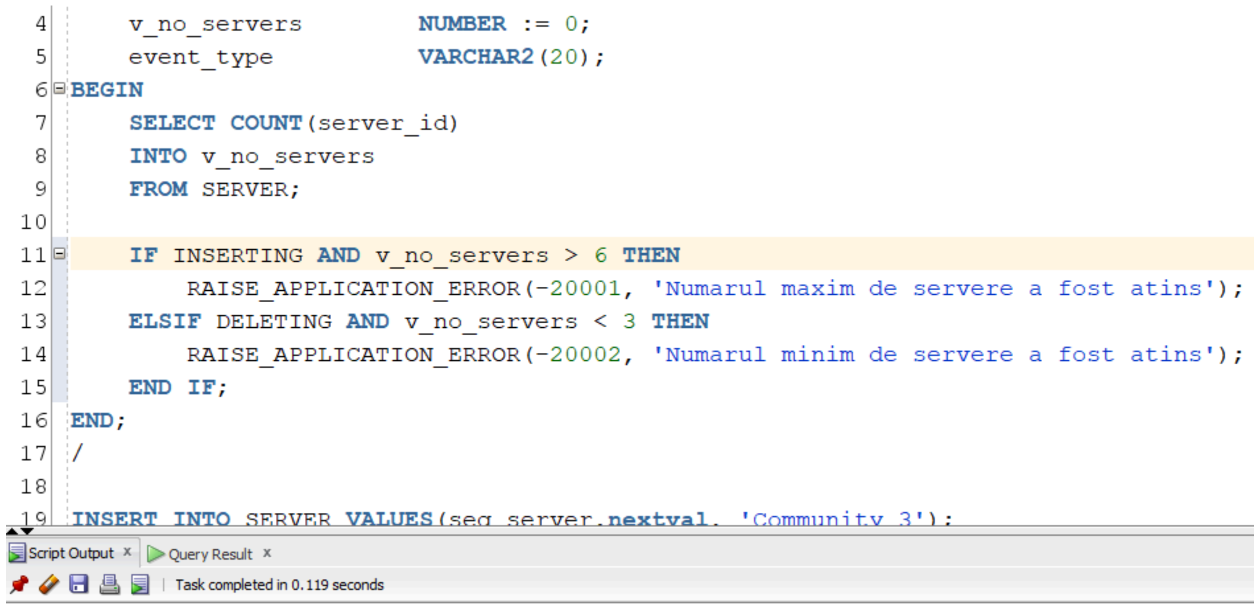
```
Task completed in 0.111 seconds

PL/SQL procedure successfully completed.
```

```
Wasteland Desert
Knife
Nu exista jucatori in biome-ul gasit care sa detine item-ul de putere maxima
```

10. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.

Scrieți un trigger care nu permite inserarea de date în tabelul SERVER dacă numărul de servere depășește 6 și nu permite ștergerea dacă numărul de servere scade sub 3 (menține numărul de servere între 2 și 7).

```
CREATE OR REPLACE TRIGGER max_servers
    AFTER INSERT OR DELETE ON SERVER
DECLARE
    v_no_servers        NUMBER := 0;
    event_type          VARCHAR2(20);
BEGIN
    SELECT COUNT(server_id)
    INTO v_no_servers
    FROM SERVER;
    IF INSERTING AND v_no_servers > 6 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Numarul maxim de
servere a fost atins');
    ELSIF DELETING AND v_no_servers < 3 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Numarul minim de
servere a fost atins');
    END IF;
END;
/
```

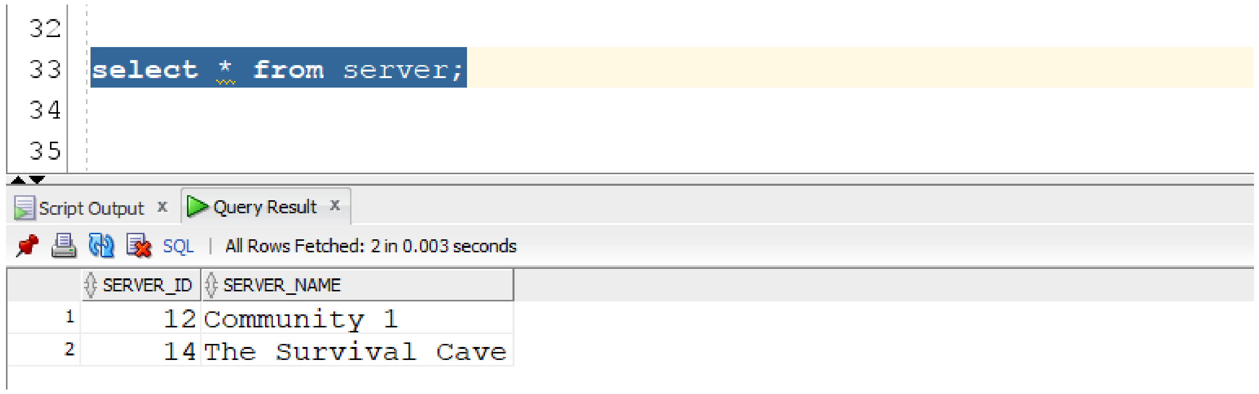


```
    v_no_servers         NUMBER := 0;
    event_type           VARCHAR2(20);
BEGIN
    SELECT COUNT(server_id)
    INTO v_no_servers
    FROM SERVER;

    IF INSERTING AND v_no_servers > 6 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Numarul maxim de servere a fost atins');
    ELSIF DELETING AND v_no_servers < 3 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Numarul minim de servere a fost atins');
    END IF;
END;
/

INSERT INTO SERVER VALUES(seq_server.nextval, 'Community 3');
```

Task completed in 0.119 seconds

```
Trigger MAX_SERVERS compiled
```

Tabelul SERVER înainte de inserări sau delete-uri

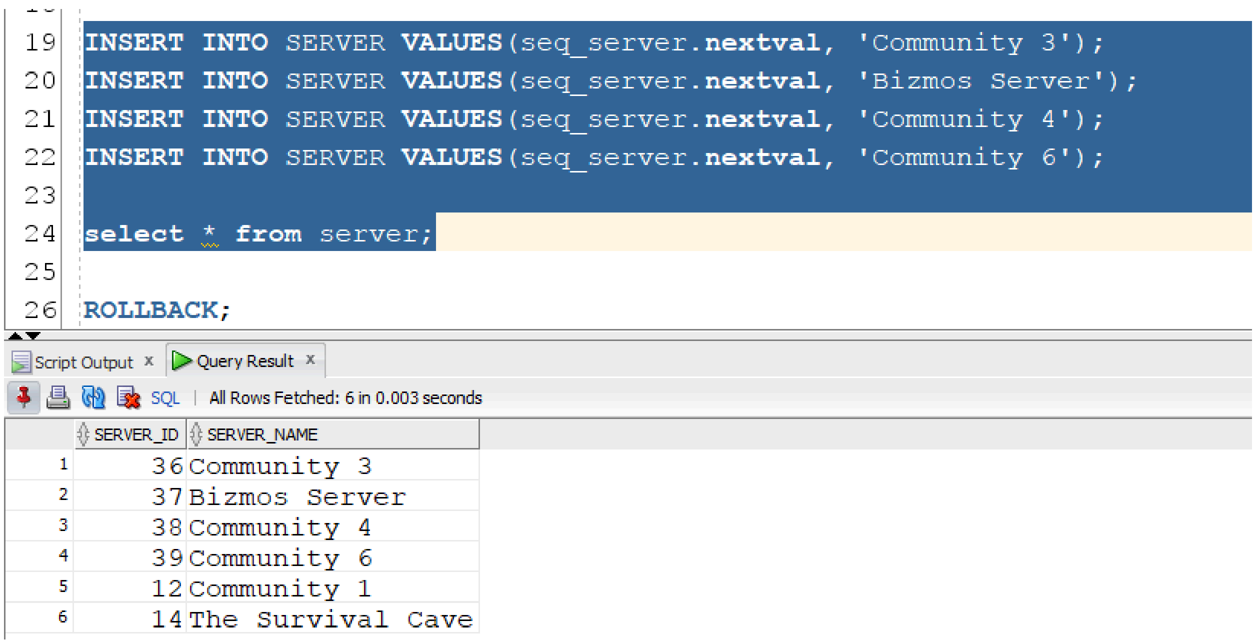


```
select * from server;
```

All Rows Fetched: 2 in 0.003 seconds

	SERVER_ID	SERVER_NAME
1	12	Community 1
2	14	The Survival Cave

Tabelul după ce am inserat încă 4 servere (încă se încadrează în limită)



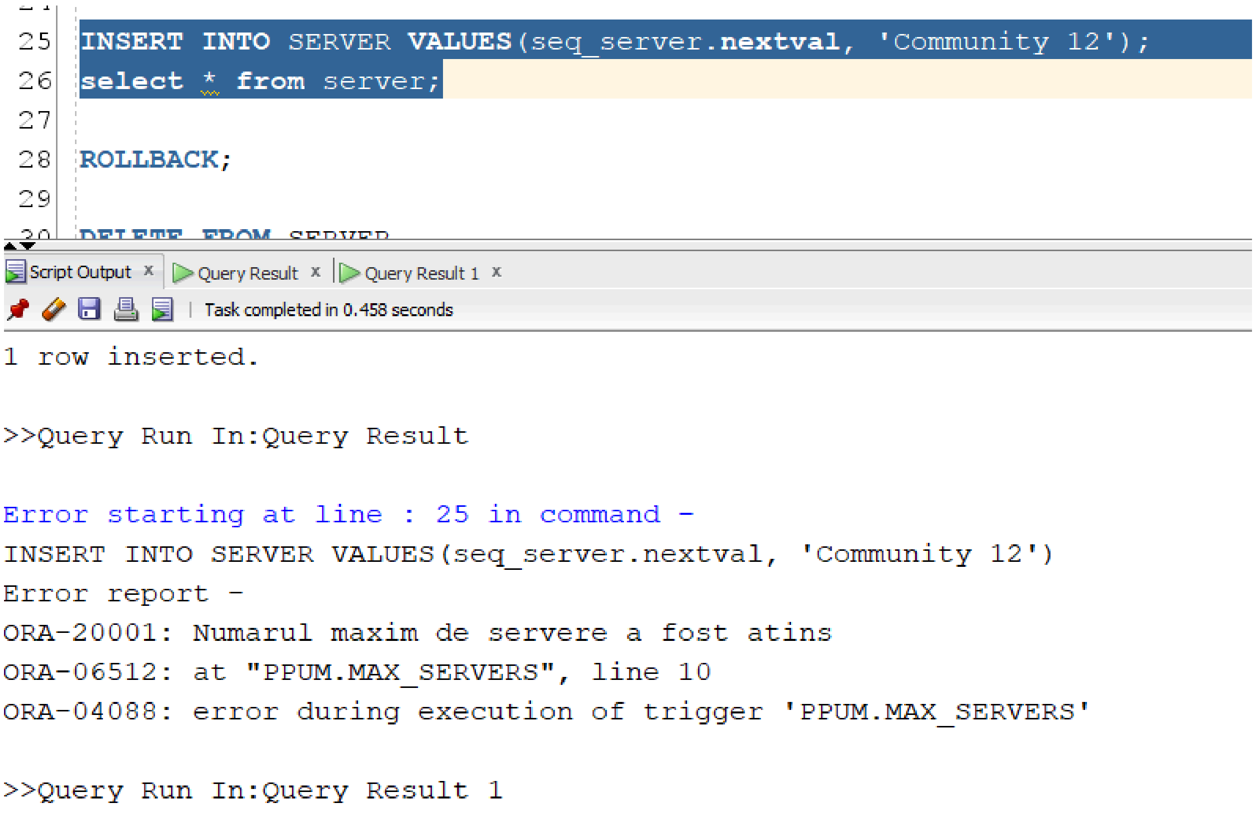
```
INSERT INTO SERVER VALUES(seq_server.nextval, 'Community 3');
INSERT INTO SERVER VALUES(seq_server.nextval, 'Bizmos Server');
INSERT INTO SERVER VALUES(seq_server.nextval, 'Community 4');
INSERT INTO SERVER VALUES(seq_server.nextval, 'Community 6');

select * from server;

ROLLBACK;
```

Script Output | Query Result

All Rows Fetched: 6 in 0.003 seconds

	SERVER_ID	SERVER_NAME
1	36	Community 3
2	37	Bizmos Server
3	38	Community 4
4	39	Community 6
5	12	Community 1
6	14	The Survival Cave

Trigger-ul se activează când încercăm să mai adăugăm înregistrări



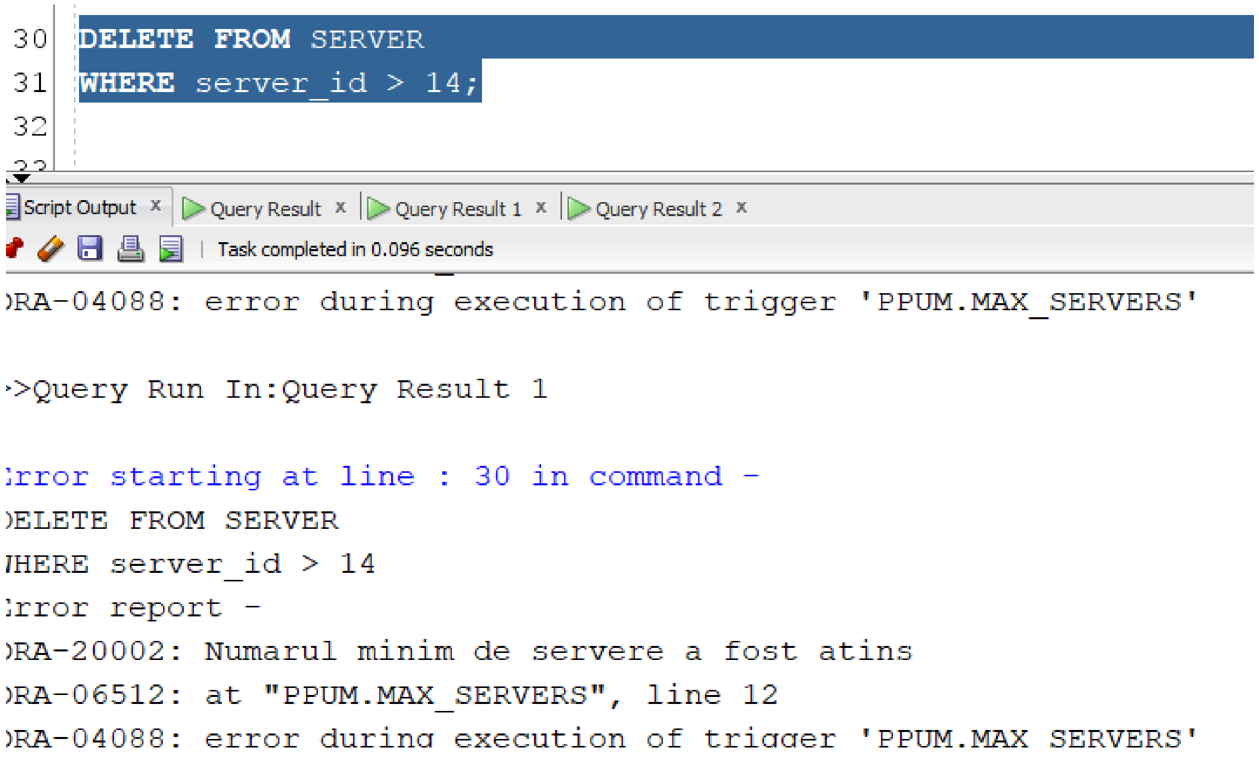
```
INSERT INTO SERVER VALUES(seq_server.nextval, 'Community 12');
select * from server;

ROLLBACK;
```

Script Output | Query Result | Query Result 1

Task completed in 0.458 seconds

```
1 row inserted.

>>Query Run In:Query Result

Error starting at line : 25 in command -
INSERT INTO SERVER VALUES(seq_server.nextval, 'Community 12')
Error report -
ORA-20001: Numarul maxim de servere a fost atins
ORA-06512: at "PPUM.MAX_SERVERS", line 10
ORA-04088: error during execution of trigger 'PPUM.MAX_SERVERS'

>>Query Run In:Query Result 1
```

Încercăm să ștergem 4 servere. Trigger-ul se activează pentru delete, pentru că numărul serverelor ar scădea sub 3. Delete-ul nu se aplică pentru că o eroare a fost ridicată în trigger



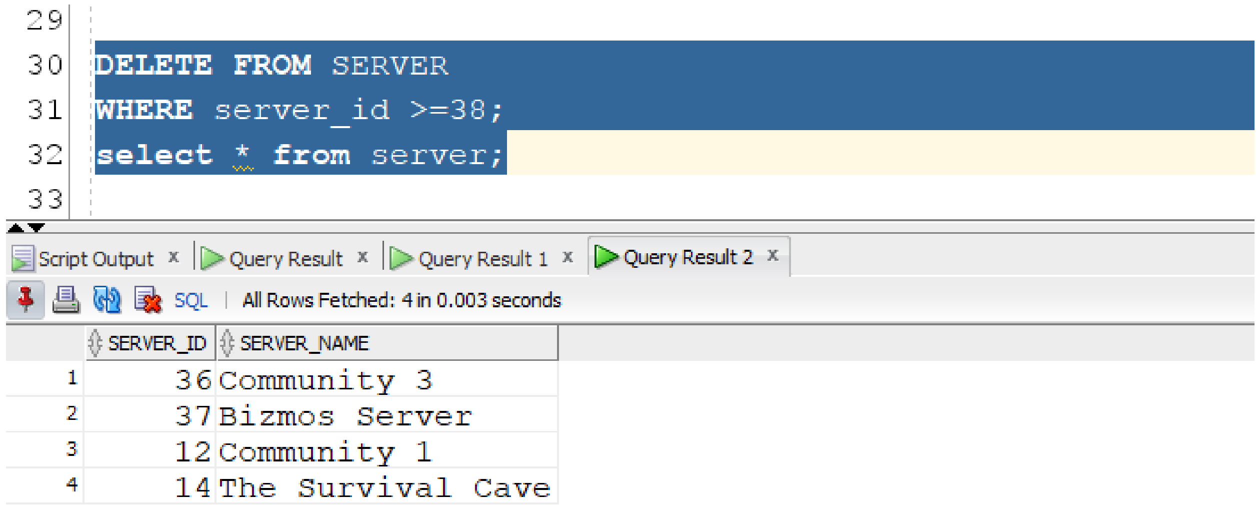
```
DELETE FROM SERVER
WHERE server_id > 14;
```

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

Task completed in 0.096 seconds

```
)RA-04088: error during execution of trigger 'PPUM.MAX_SERVERS'

›>Query Run In:Query Result 1

:rror starting at line : 30 in command -
)ELETE FROM SERVER
/HERE server_id > 14
:rror report -
)RA-20002: Numarul minim de servere a fost atins
)RA-06512: at "PPUM.MAX_SERVERS", line 12
)RA-04088: error during execution of trigger 'PPUM.MAX_SERVERS'
```

Tabelul după ce am șters 2 înregistrări. Trigger-ul nu se activează pentru că numărul de servere rămâne mai mare decât 2 și mai mic decât 7.



```
DELETE FROM SERVER
WHERE server_id >=38;
select * from server;
```

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

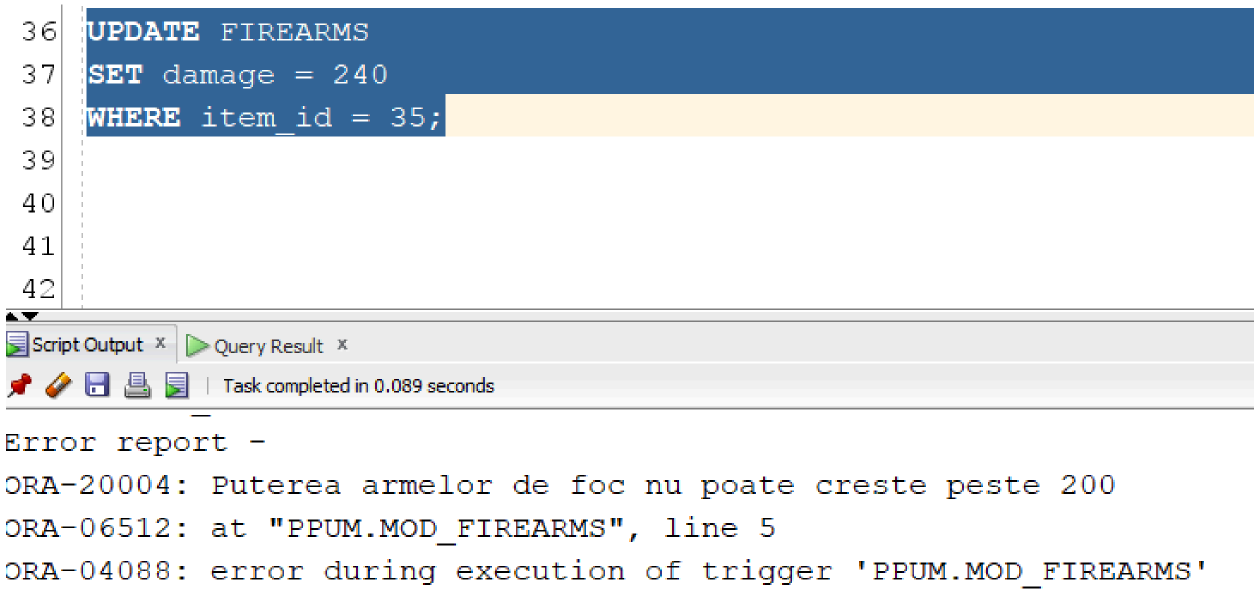
SQL | All Rows Fetched: 4 in 0.003 seconds

	SERVER_ID	SERVER_NAME
1	36	Community 3
2	37	Bizmos Server
3	12	Community 1
4	14	The Survival Cave

11. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.

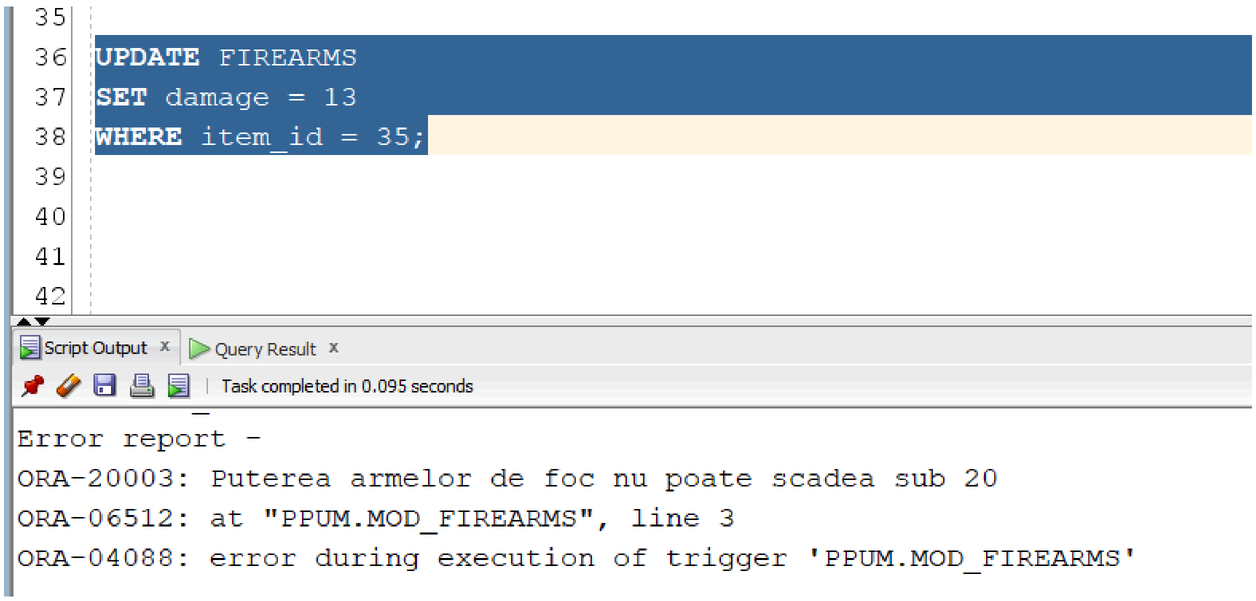
Scrieți un trigger care nu permite modificarea puterii unei arme de foc sub 20 sau peste 200.

```
CREATE OR REPLACE TRIGGER mod_firearms
    BEFORE UPDATE OF damage ON FIREARMS
    FOR EACH ROW
BEGIN
    IF(:NEW.damage < 20) THEN
        RAISE_APPLICATION_ERROR(-20003, 'Puterea armelor de foc
nu poate scadea sub 20');
    ELSIF(:NEW.damage > 200) THEN
        RAISE_APPLICATION_ERROR(-20004, 'Puterea armelor de foc
nu poate creste peste 200');
    END IF;
END;
/
```



```
UPDATE FIREARMS
SET damage = 240
WHERE item_id = 35;
```

```
Error report -
ORA-20004: Puterea armelor de foc nu poate creste peste 200
ORA-06512: at "PPUM.MOD_FIREARMS", line 5
ORA-04088: error during execution of trigger 'PPUM.MOD_FIREARMS'
```

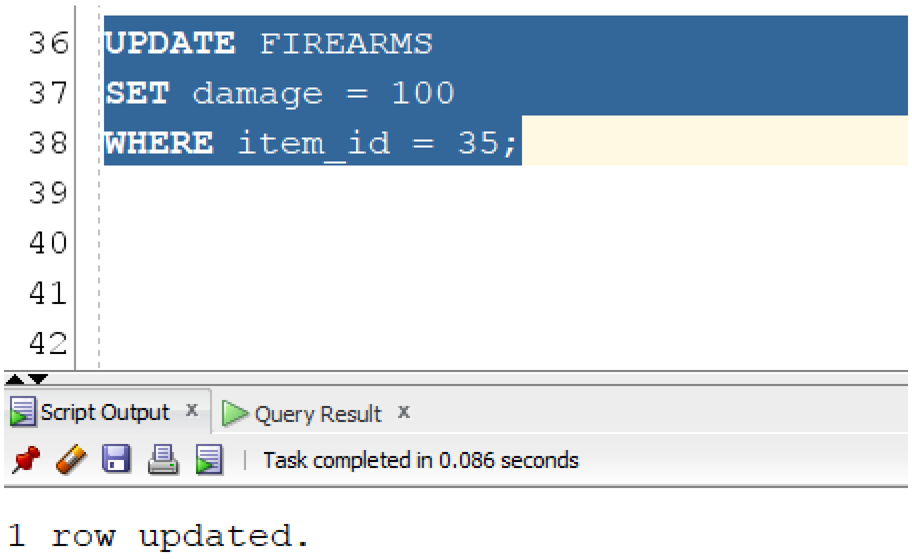


```
UPDATE FIREARMS
SET damage = 13
WHERE item_id = 35;
```

Script Output × Query Result ×

Task completed in 0.095 seconds

```
Error report -
ORA-20003: Puterea armelor de foc nu poate scadea sub 20
ORA-06512: at "PPUM.MOD_FIREARMS", line 3
ORA-04088: error during execution of trigger 'PPUM.MOD_FIREARMS'
```



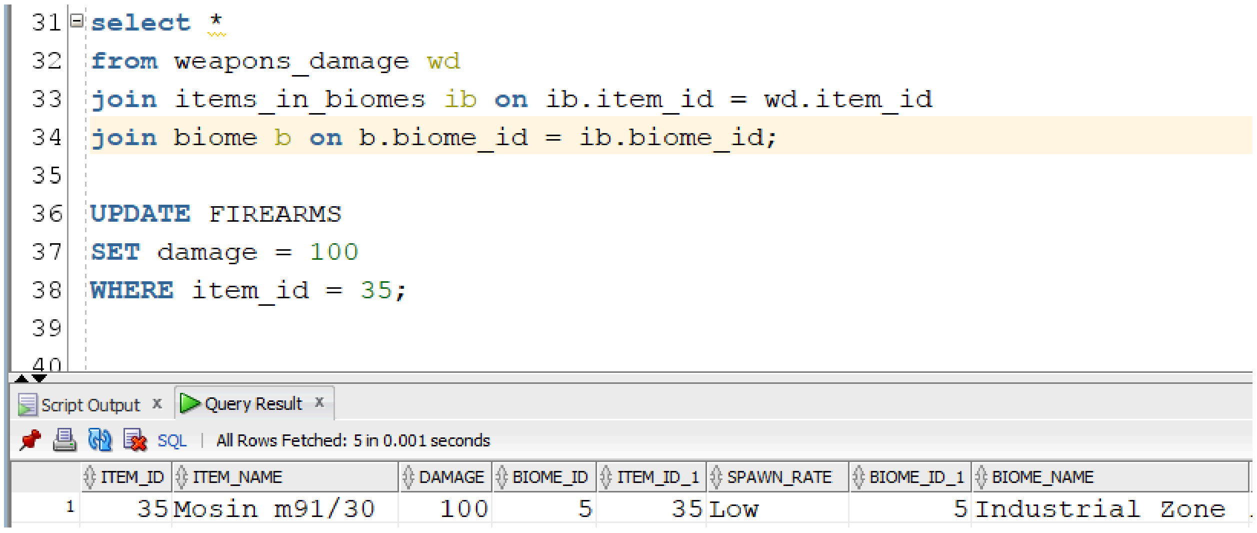
```
UPDATE FIREARMS
SET damage = 100
WHERE item_id = 35;
```

Script Output × Query Result ×

Task completed in 0.086 seconds

```
1 row updated.
```

Schimbările se pot vedea în acest tabel (item-ul cu id 35 are damage 100 dupa UPDATE)



```
select *
from weapons_damage wd
join items_in_biomes ib on ib.item_id = wd.item_id
join biome b on b.biome_id = ib.biome_id;

UPDATE FIREARMS
SET damage = 100
WHERE item_id = 35;
```

All Rows Fetched: 5 in 0.001 seconds

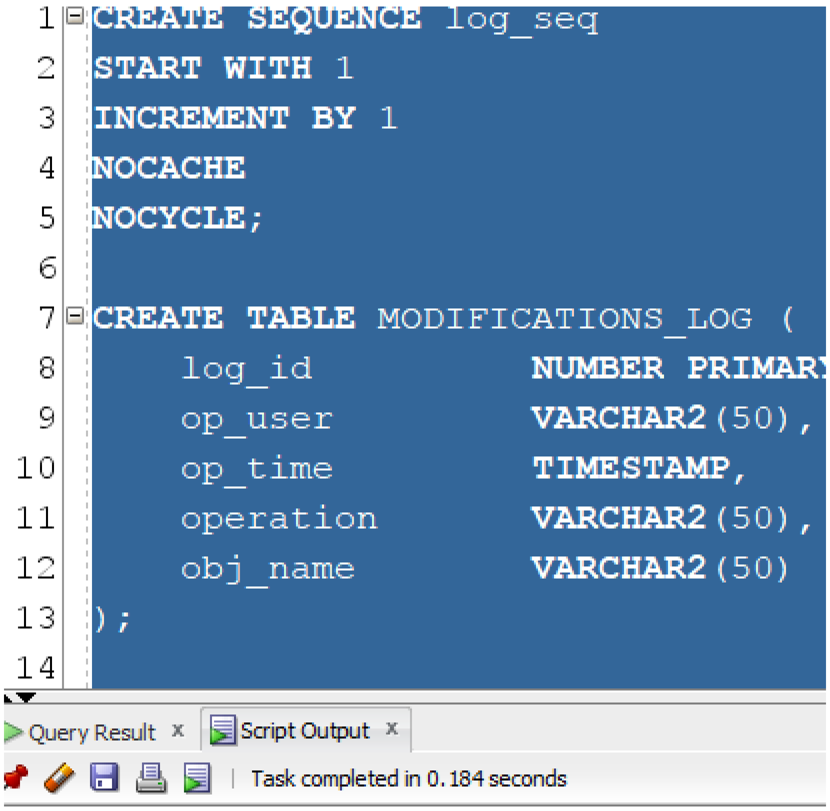
	ITEM_ID	ITEM_NAME	DAMAGE	BIOME_ID	ITEM_ID_1	SPAWN_RATE	BIOME_ID_1	BIOME_NAME
1	35	Mosin m91/30	100	5	35	Low	5	Industrial Zone

12. Definiți un trigger de tip LDD. Declanșați trigger-ul.

Scrieți un trigger LDD care se activează când se fac operații LDD pe baza de date. Trigger-ul inregistrează într-o tabela schimbările făcute de user-ul PPUM și ridică o eroare dacă un alt user încearcă să facă schimbări LDD.

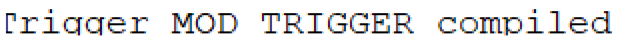
```
CREATE SEQUENCE log_seq
START WITH 1
INCREMENT BY 1
NOCACHE
NOCYCLE;

CREATE TABLE MODIFICATIONS_LOG (
    log_id          NUMBER PRIMARY KEY,
    op_user         VARCHAR2(50),
    op_time         TIMESTAMP,
    operation       VARCHAR2(50),
    obj_name        VARCHAR2(50)
);
```

```
CREATE OR REPLACE TRIGGER MOD_TRIGGER
    BEFORE DROP OR CREATE OR ALTER ON SCHEMA
DECLARE
    v_user      VARCHAR2(100);
BEGIN
    SELECT USER INTO v_user FROM DUAL;

    IF UPPER(v_user) = 'PPUM' THEN
        INSERT INTO MODIFICATIONS_LOG(log_id, op_user, op_time,
operation, obj_name)
        VALUES (log_seq.NEXTVAL, v_user, SYSTIMESTAMP,
sys.sysevent, sys.dictionary_obj_name);
    ELSE
        RAISE_APPLICATION_ERROR(-20005, 'Operatie LDD
neautorizata facuta de ' || v_user);
    END IF;
END;
/
```



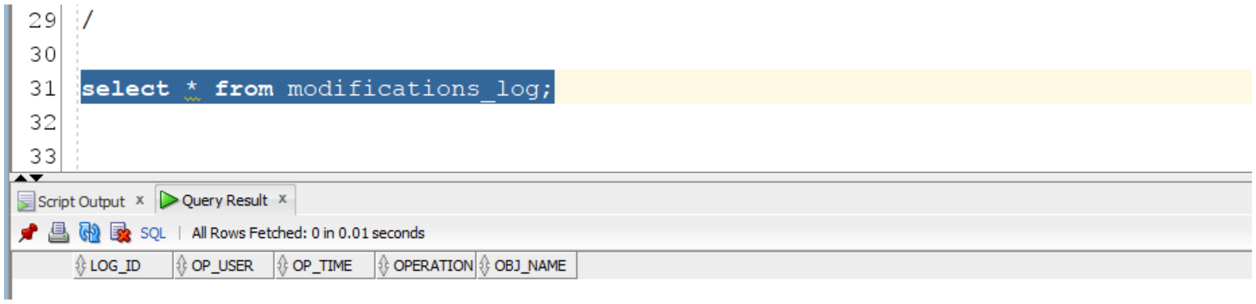
```
CREATE SEQUENCE log_seq
START WITH 1
INCREMENT BY 1
NOCACHE
NOCYCLE;

CREATE TABLE MODIFICATIONS_LOG (
    log_id          NUMBER PRIMARY
    op_user         VARCHAR2(50),
    op_time         TIMESTAMP,
    operation       VARCHAR2(50),
    obj_name        VARCHAR2(50)
);

```

```
Task completed in 0.184 seconds
```



```
Trigger MOD TRIGGER compiled
```

Tabela de modificări înainte de declanșarea trigger-ului

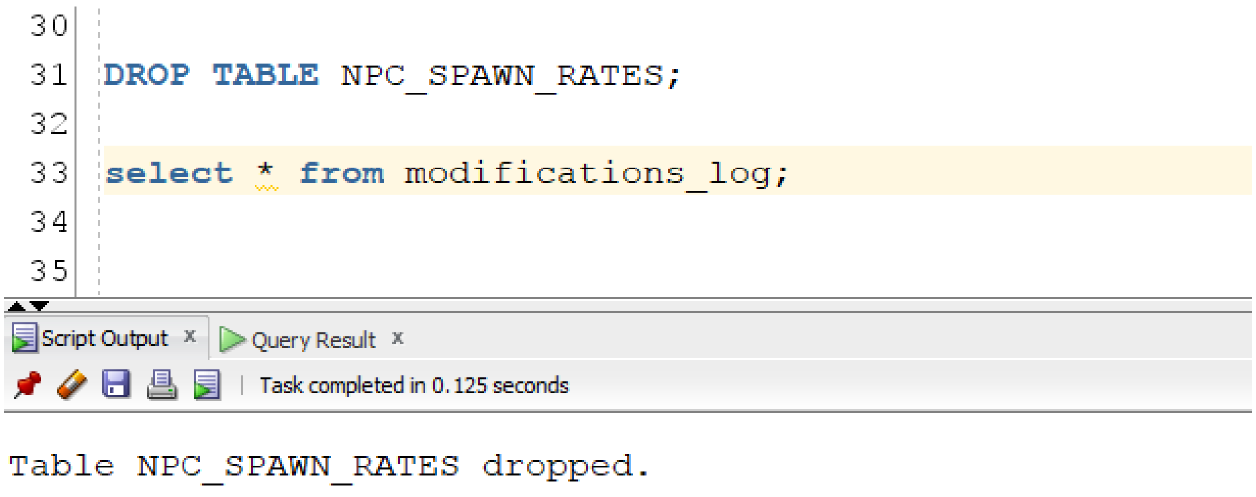


```
/

select * from modifications_log;
```

```
All Rows Fetched: 0 in 0.01 seconds
```

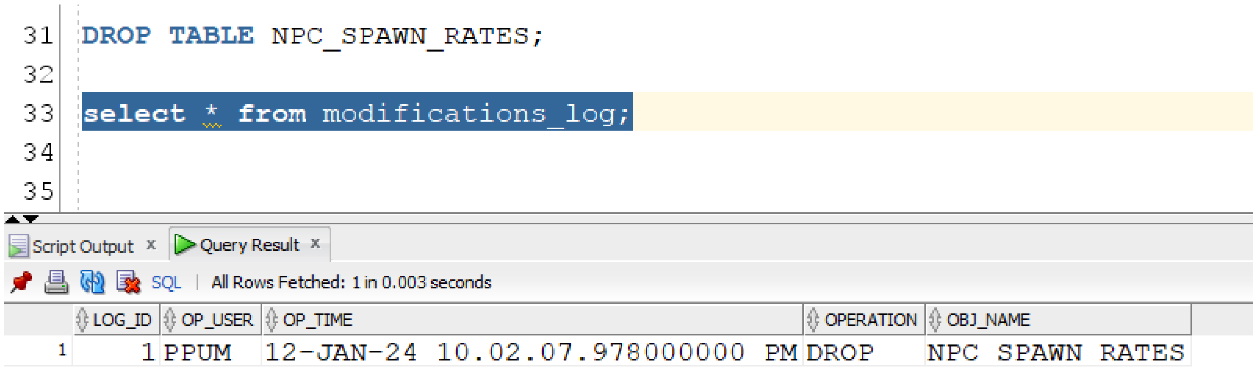
LOG_ID	OP_USER	OP_TIME	OPERATION	OBJ_NAME

Ștergem tabela NPC_SPAWN_RATES


```
DROP TABLE NPC_SPAWN_RATES;

select * from modifications_log;
```

```
Task completed in 0.125 seconds

Table NPC_SPAWN_RATES dropped.
```

Trigger-ul se activează și putem vedea datele introduse în tabelă


```
DROP TABLE NPC_SPAWN_RATES;

select * from modifications_log;
```

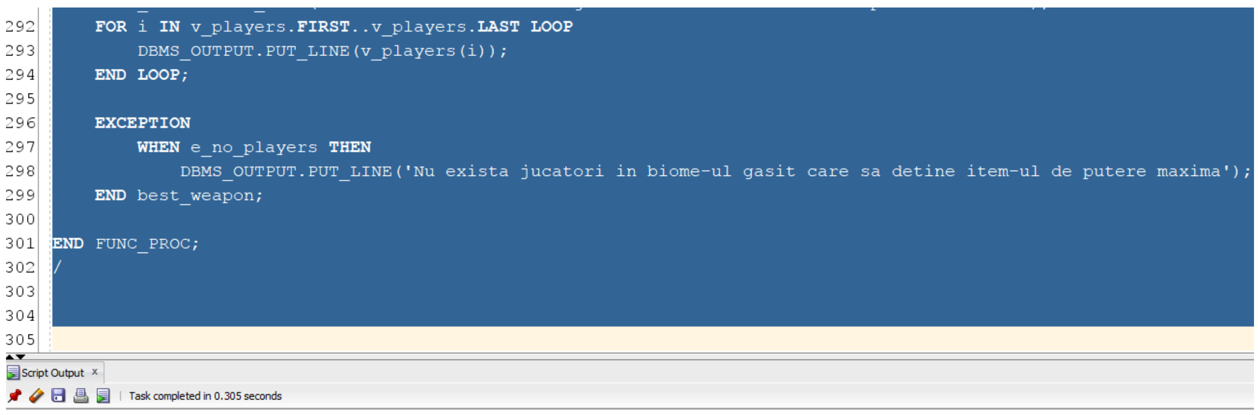
```
All Rows Fetched: 1 in 0.003 seconds
```

	LOG_ID	OP_USER	OP_TIME	OPERATION	OBJ_NAME
1	1	PPUM	12-JAN-24 10.02.07.978000000 PM	DROP	NPC_SPAWN_RATES

13. Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului.

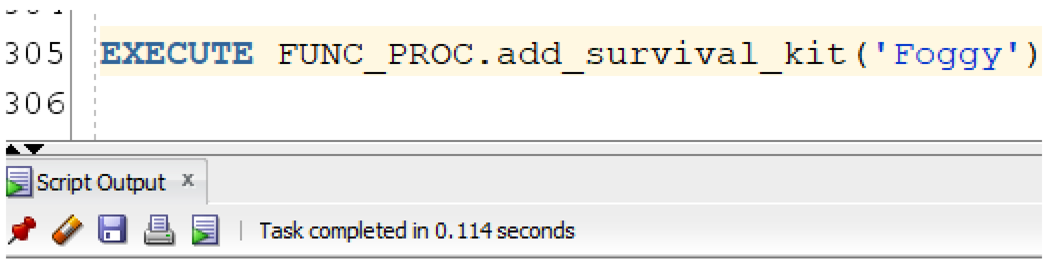
```
CREATE OR REPLACE PACKAGE FUNC_PROC AS
    PROCEDURE       add_survival_kit(w_name
WEATHER.weather_name%TYPE);
    PROCEDURE       p_power(sv_name  SERVER.server_name%TYPE);
    FUNCTION        player_health_data(p_name
PLAYER.player_name%TYPE)
                    RETURN VARCHAR2;
    PROCEDURE       best_weapon(n_name NPC.npc_name%TYPE,
    s_rate NPC_IN_BIOMES.spawn_rate%TYPE);
END FUNC_PROC;
/

CREATE OR REPLACE PACKAGE BODY FUNC_PROC AS
    --6
    PROCEDURE add_survival_kit
    (w_name WEATHER.weather_name%TYPE)
    IS
        --tablou indexat pentru a retine id-urile jucatorilor
        TYPE player_ids IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
        --tablou imbricat care retine itemele din kit
        TYPE kit_items IS TABLE OF NUMBER;
        --vector pentru a retine cate iteme de fiecare tip
        --sunt adaugate
        TYPE items_vect IS VARRAY(10) OF NUMBER;

        CURSOR med_items(pid PLAYER.player_id%TYPE) IS
            SELECT it.item_id
            FROM ITEM it
            JOIN CONTAINS c on c.item_id = it.item_id
            JOIN INVENTORY i on i.inventory_id = c.inventory_id
            WHERE i.player_id = pid AND it.item_type =
'Medical';

        qnt         items_vect := items_vect(1, 1, 2, 1, 1);
        itms        kit_items;
        p_ids       player_ids;
        v_inv_id    CONTAINS.inventory_id%TYPE;
        v_exists    BOOLEAN;
    BEGIN
        --selectam jucatorii care se afla in locatii
        --cu vremea primita ca argument
        SELECT PLAYER_ID BULK COLLECT
        INTO p_ids
        FROM (SELECT L.PLAYER_ID
              FROM LOCATION L
              JOIN WEATHER W ON W.weather_id = L.weather_id
              WHERE UPPER(W.weather_name) = UPPER(w_name)
            );
        --selectam itemele
        SELECT item_id BULK COLLECT
        INTO itms
        FROM ITEM
        WHERE ITEM.item_type = 'Medical';

        --adaugam itemele in inventarele jucatorilor
        FOR i IN p_ids.FIRST..p_ids.LAST LOOP
            SELECT inventory_id INTO v_inv_id
            FROM INVENTORY
            WHERE PLAYER_ID = p_ids(i);

            FOR j IN itms.FIRST..itms.LAST LOOP
                v_exists := FALSE;

                FOR k IN med_items(p_ids(i)) LOOP
                    IF k.item_id = itms(j) THEN
                        v_exists := TRUE;
                        EXIT;
                    END IF;
                END LOOP;

                --daca exista unul din iteme in inventar
                IF v_exists THEN
                    CONTINUE;
                END IF;

                INSERT INTO CONTAINS VALUES(v_inv_id, itms(j),
qnt(j));
            END LOOP;
        END LOOP;
    END add_survival_kit;

    --7
    PROCEDURE p_power
    (sv_name  SERVER.server_name%TYPE)
    IS
        pwr        FIREARMS.damage%TYPE;

        CURSOR ppower(pid PLAYER.player_id%TYPE) IS
            SELECT damage
            FROM FIREARMS f
            JOIN ITEM it ON it.item_id = f.item_id
            JOIN CONTAINS c ON c.item_id = it.item_id
            JOIN INVENTORY i ON i.inventory_id = c.inventory_id
            WHERE i.player_id = pid
            UNION
            SELECT damage
            FROM MELEE_WEAPONS mw
            JOIN ITEM it ON it.item_id = mw.item_id
            JOIN CONTAINS c ON c.item_id = it.item_id
            JOIN INVENTORY i ON i.inventory_id = c.inventory_id
            WHERE i.player_id = pid;

        v_pow               FIREARMS.damage%TYPE;
        v_no_weapons        NUMBER;
        v_pid               PLAYER.player_id%TYPE;
        v_pname             PLAYER.player_name%TYPE;
        v_pl                NUMBER;
    BEGIN
        v_pl := 0;
        pwr := 0;
        FOR i IN (  SELECT p.player_name, p.player_id
                    FROM PLAYER p
                    JOIN SERVER_CONNECTIONS sc ON sc.player_id =
p.player_id
                    JOIN SERVER s ON s.server_id = sc.server_id
                    WHERE s.server_name = sv_name
                        )LOOP
            v_pow := 0;
            v_no_weapons := 0;
            v_pl := v_pl + 1;
            FOR j IN ppower(i.player_id) LOOP
                v_pow := v_pow + j.damage;
                v_no_weapons := v_no_weapons + 1;
            END LOOP;
        pwr := pwr + v_pow;
        END LOOP;
        IF v_pl = 0 THEN
            RAISE_APPLICATION_ERROR(-20000, 'Serverul nu are
jucatori!');
        END IF;
        DBMS_OUTPUT.PUT_LINE('Puterea totala a jucatorilor de pe
' ||
        sv_name || ' este ' || pwr);
    END p_power;


    --8
    FUNCTION player_health_data
    (p_name PLAYER.player_name%TYPE)
    RETURN VARCHAR2 IS

    CURSOR player_medical(pid PLAYER.player_id%TYPE) IS
        SELECT m.health_effect
        FROM MEDICAL m
        JOIN ITEM it ON it.item_id = m.item_id
        JOIN CONTAINS c ON c.item_id = it.item_id
        JOIN INVENTORY i ON i.inventory_id = c.inventory_id
        WHERE i.player_id = pid;

    v_medical           MEDICAL.health_effect%TYPE;
    --numarul de iteme medicale din inventar
    v_medical_it        NUMBER;
    v_pid               PLAYER.player_id%TYPE;
    v_name              PLAYER.player_name%TYPE;
    v_health_total      STATS.health%TYPE;
    v_avg_healing       MEDICAL.health_effect%TYPE;
    --numarul de minute supravietuite
    v_min               NUMBER;
    v_ret               VARCHAR2(25);

    --exceptii
    e_player_dead               EXCEPTION;
    e_player_epi_only           EXCEPTION;
    e_player_no_medical         EXCEPTION;
    e_player_not_found          EXCEPTION;

    BEGIN
        v_medical := 0;
        v_medical_it := 0;

        --calculam media abilitatilor medicale
        --ale jucatorilor
        SELECt avg(health_ability)
        INTO v_avg_healing
        FROM(
            SELECT p.player_id, sum(m.health_effect) as
health_ability
            FROM player p
            JOIN inventory i on i.player_id = p.player_id
            JOIN contains c on c.inventory_id = i.inventory_id
            JOIN medical m on m.item_id = c.item_id
            GROUP BY p.player_id
            );

        --gasim id-ul si numele jucatorului
        SELECT p.player_id, p.player_name
        INTO v_pid, v_name
        FROM PLAYER p
        WHERE UPPER(p.player_name) LIKE UPPER(p_name);

        --gasim viata initiala a jucatorului
        SELECT s.health
        INTO v_health_total
        FROM STATS s
        WHERE s.player_id = v_pid;

        --calculam abilitatea medicala a jucatorului
        FOR i IN player_medical(v_pid) LOOP
            v_medical := v_medical + i.health_effect;
            v_medical_it := v_medical_it + 1;
        END LOOP;

        IF v_health_total = 0 THEN
            RAISE e_player_dead;
        END IF;

        v_health_total := v_health_total + v_medical;
        v_min := FLOOR(v_health_total / 15);

        IF v_medical < v_avg_healing THEN
            v_ret := 'sub medie';
        ELSIF v_medical = v_avg_healing THEN
```

```
            v_ret := 'medii';
        ELSE
            v_ret := 'peste medie';
        END IF;

        IF v_medical = 0 AND v_medical_it = 0 THEN
            RAISE e_player_no_medical;
        ELSIF v_medical = 0 AND v_medical_it <> 0 THEN
            RAISE e_player_epi_only;
        END IF;

        --returnam info
        RETURN 'Jucatorul ' || v_name || ' are abilitatea
medicala ' ||
                v_medical || ' si poate rezista ' || v_min || '
minute.' ||
                ' Jucatorul are abilitati medicale ' || v_ret;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RETURN 'Nu a fost gasit un jucator cu numele dat';
        WHEN e_player_no_medical THEN
            RETURN 'Jucatorul nu are iteme medicale';
        WHEN e_player_epi_only THEN
            RETURN 'Jucatorul are doar EpiPen-uri in inventar';
        WHEN e_player_dead THEN
            RETURN 'Personajul jucatorului nu mai este in
viata';
    END player_health_data;


    --9
    PROCEDURE best_weapon
    (n_name NPC.npc_name%TYPE,
    s_rate NPC_IN_BIOMES.spawn_rate%TYPE) IS

    TYPE players_vect IS VARRAY(25) OF VARCHAR2(50);

    v_biome_name        BIOME.biome_name%TYPE;
    v_item_id           ITEM.item_id%TYPE;
    v_item_name         ITEM.item_name%TYPE;
    v_item_dmg          FIREARMS.damage%TYPE;
    v_players           players_vect := players_vect();

    e_no_players        EXCEPTION;

    BEGIN
    --gasim biome-ul in care npc-ul dat se spawneaza
    --cu spawn rate-ul dat
    BEGIN
        SELECT b.biome_name
        INTO v_biome_name
        FROM BIOME b
        JOIN NPC_IN_BIOMES nb on nb.biome_id = b.biome_id
        JOIN NPC n on n.npc_id = nb.npc_id
        WHERE UPPER(n.npc_name) LIKE UPPER(n_name)
            AND UPPER(nb.spawn_rate) LIKE UPPER(s_rate);
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('NPC-ul dat nu se spawneaza in
niciun biome cu spawn rate-ul dat.');
            RETURN; --iesim din procedura
        WHEN TOO_MANY_ROWS THEN
            DBMS_OUTPUT.PUT_LINE('Npc-ul dat se spawneaza cu
spawn rate-ul dat in mai multe biome-uri.');
            RETURN;
    END;

    DBMS_OUTPUT.PUT_LINE(v_biome_name);

    --cautam arma cu cel mai mult damage
    --din biome-ul gasit
    BEGIN
        SELECT wd.item_id, wd.item_name
        INTO v_item_id, v_item_name
        FROM weapons_damage wd
        JOIN ITEMS_IN_BIOMES ib on ib.item_id = wd.item_id
        JOIN BIOME b on b.biome_id = ib.biome_id
        WHERE UPPER(b.biome_name) LIKE UPPER(v_biome_name)
        AND wd.damage = (SELECT MAX(wd.damage)
                         FROM weapons_damage wd
                         JOIN ITEMS_IN_BIOMES ib on ib.item_id =
wd.item_id
                         JOIN BIOME b on b.biome_id =
ib.biome_id
                         WHERE UPPER(b.biome_name) LIKE
UPPER(v_biome_name)
                         );
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('Nu se spawneaza arme in
biome-ul gasit.');
            RETURN;
        WHEN TOO_MANY_ROWS THEN
            DBMS_OUTPUT.PUT_LINE('Se spawneaza mai multe arme cu
putere maxima in biome-ul gasit.');
            RETURN;
    END;

    DBMS_OUTPUT.PUT_LINE(v_item_name);

    --verificam daca arma apare in
    --inventarul unui jucator
    SELECT p.player_name
    BULK COLLECT INTO v_players
    FROM PLAYER p
    JOIN INVENTORY i on i.player_id = p.player_id
    JOIN CONTAINS C on c.inventory_id = i.inventory_id
    JOIN LOCATION l on l.player_id = p.player_id
    JOIN BIOME B on b.biome_id = l.biome_id
    WHERE UPPER(b.biome_name) LIKE UPPER(v_biome_name)
    AND c.item_id = v_item_id;

    IF v_players.COUNT = 0 THEN
        RAISE e_no_players;
    END IF;

    DBMS_OUTPUT.PUT_LINE('Jucatorii din biome-ul gasit care
detin item-ul de putere maxima:');
    FOR i IN v_players.FIRST..v_players.LAST LOOP
        DBMS_OUTPUT.PUT_LINE(v_players(i));
    END LOOP;
    EXCEPTION
        WHEN e_no_players THEN
            DBMS_OUTPUT.PUT_LINE('Nu exista jucatori in biome-ul
gasit care sa detine item-ul de putere maxima');
    END best_weapon;

END FUNC_PROC;
/
```



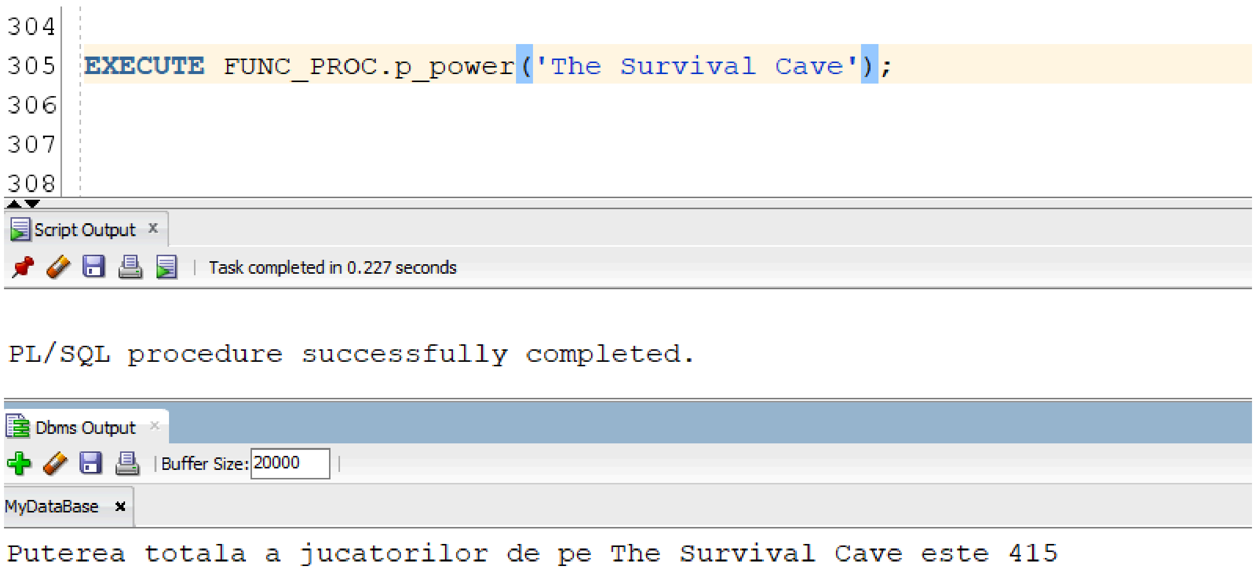
```
    FOR i IN v_players.FIRST..v_players.LAST LOOP
        DBMS_OUTPUT.PUT_LINE(v_players(i));
    END LOOP;

    EXCEPTION
        WHEN e_no_players THEN
            DBMS_OUTPUT.PUT_LINE('Nu exista jucatori in biome-ul gasit care sa detine item-ul de putere maxima');
    END best_weapon;

END FUNC_PROC;
/
```

```
Package FUNC_PROC compiled


Package Body FUNC_PROC compiled
```



```
EXECUTE FUNC_PROC.add_survival_kit('Foggy')
```

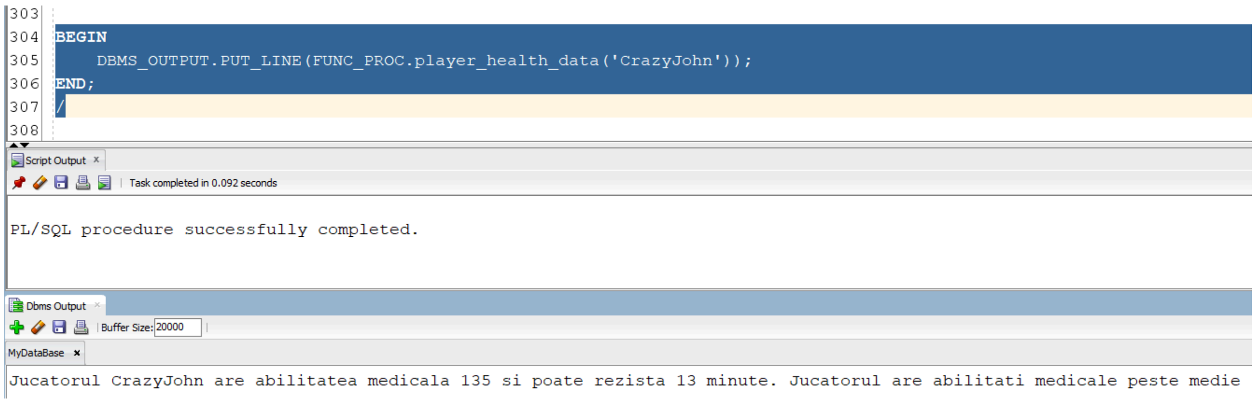
```
PL/SQL procedure successfully completed.
```



```
EXECUTE FUNC_PROC.p_power('The Survival Cave');
```

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

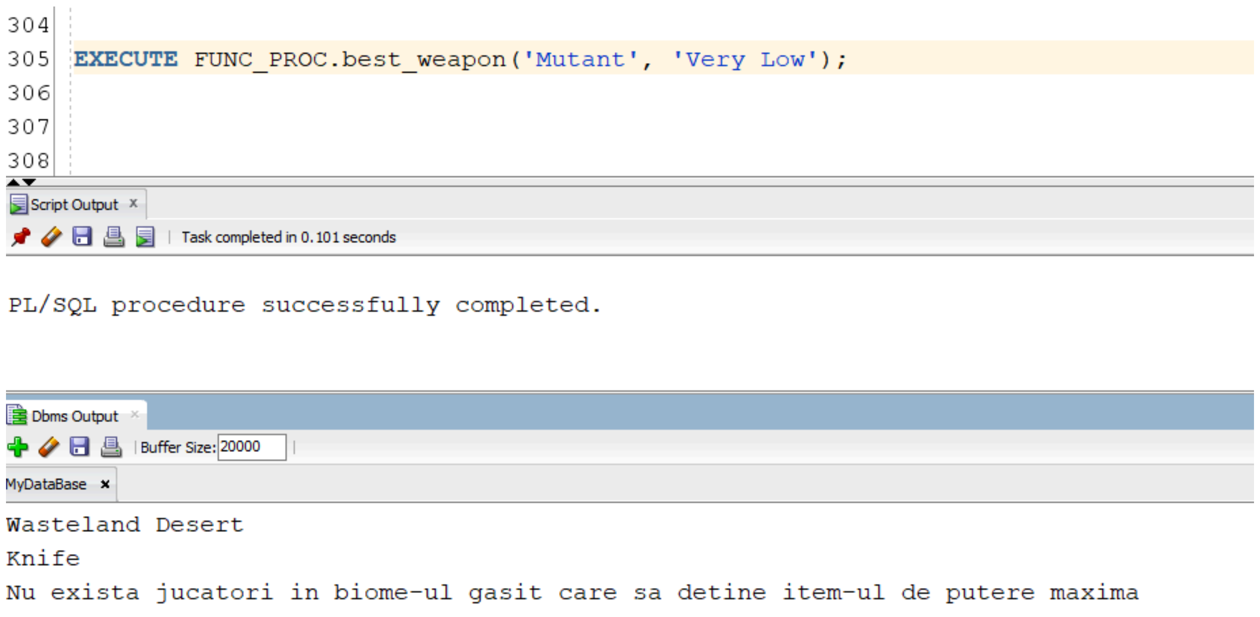
```
Puterea totala a jucatorilor de pe The Survival Cave este 415
```



```
BEGIN
    DBMS_OUTPUT.PUT_LINE(FUNC_PROC.player_health_data('CrazyJohn'));
END;
/
```

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

```
Jucatorul CrazyJohn are abilitatea medicala 135 si poate rezista 13 minute. Jucatorul are abilitati medicale peste medie
```



```
EXECUTE FUNC_PROC.best_weapon('Mutant', 'Very Low');
```

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

```
Wasteland Desert
Knife
Nu exista jucatori in biome-ul gasit care sa detine item-ul de putere maxima
```