IMAGE CRETAED WITH HTTPS://WWW.CRAIYON.COM/

Virtual_zoo.db

In a virtual zoo located in an imaginary world, a variety of fascinating species coexist in habitats carefully designed to recreate their natural environments (Habitats, name). Visitors are greeted with enthusiasm as they explore the various habitats, from lush tropical jungles to vast African plains (Visits, habitat_id). During their visit, they observe how expert caretakers attend to the dietary needs of the animals, carefully recording the feeding time, type of food, and amount consumed (Feeding, feeding_time, food_type, amount). In another corner of the zoo, breeding programs are underway to preserve endangered species, with specialists closely monitoring the progress of each program (BreedingPrograms, start_date, end_date). Among the exciting events happening at the zoo, birth records document the arrival of new offspring to the family, meticulously detailing the birth date and ancestry of each new addition (BirthRecords, birth_date, mother_id, father_id). Meanwhile, in the medical area, treatments are ongoing to ensure the health and well-being of the zoo residents, with carefully selected medications administered as needed (Treatments, start_date, end_date, medication_id). This interaction between species, habitats, feeding, breeding, and medical care creates a dynamic ecosystem where life flourishes and visitors marvel at the beauty and diversity of the animal kingdom.

Contenido

Entity-Relationship Diagram 5
Species: 6
  Columns: 6
  Definition: 6
  Example: 6
Habitats: 7
  Columns: 7
  Definition: 7
  Example: 7
Feeding: 8
  Columns: 8
  Definition: 8
  Example: 8
BreedingPrograms: 9
  Columns: 9
  Definition: 9
  Example: 9
BirthRecords: 10
  Columns: 10
  Definition: 10
  Example: 10
Visits: 11
  Columns: 11
  Definition: 11
  Example: 11
Medications: 12
  Columns: 12
  Definition: 12
  Example: 12
Treatments: 13
  Columns: 13
  Definition: 13
  Example: 13

AnimalTreatments: 14

Columns: 14

Definition: 14

Example: 14

Query simples: 15

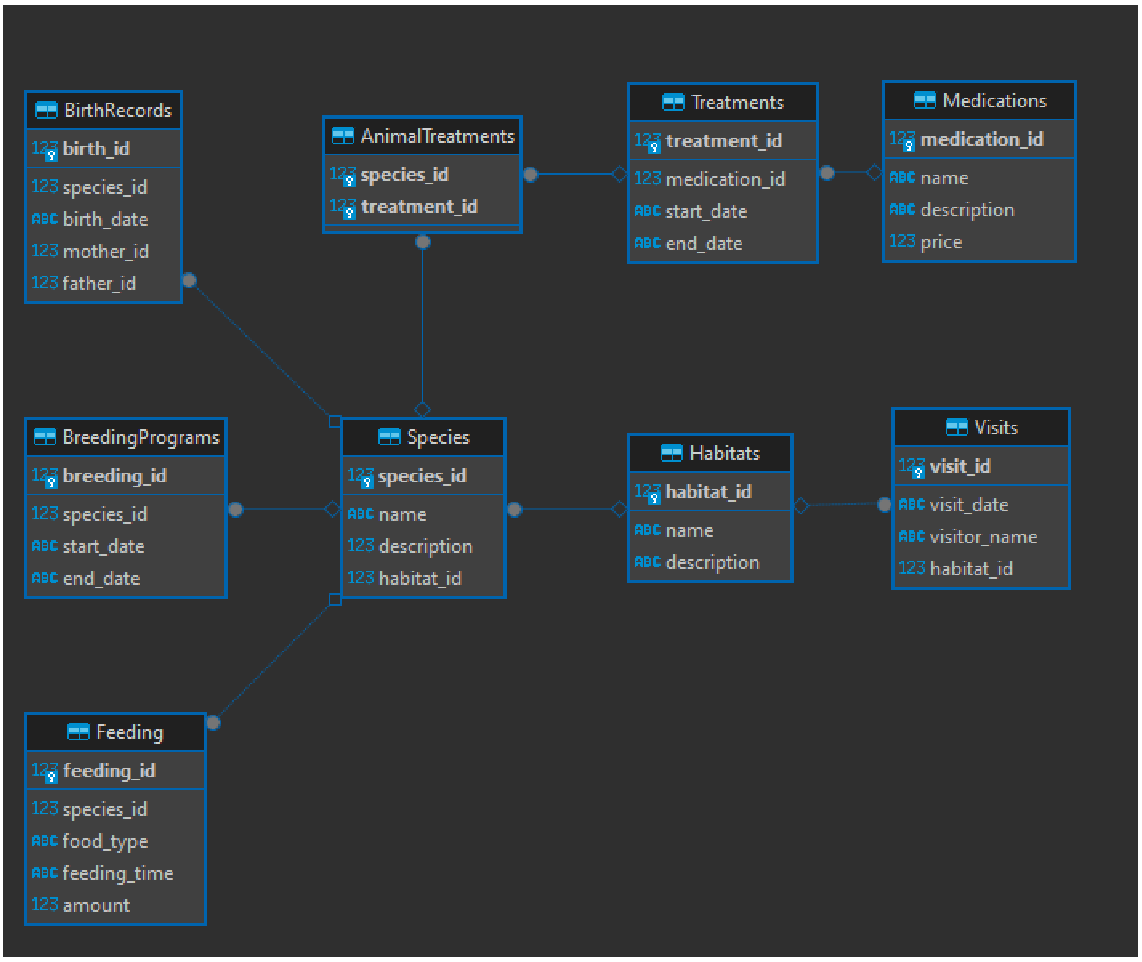
-- Get a list of all animal species along with their corresponding habitat. 15

-- Calculate the average amount of food provided to all species during a specific day. 15

-- Find all species that have received a specific treatment. 15

-- query that uses a Common Table Expression (CTE) to calculate the total treatments administered for each animal species: 15

Entity-Relationship Diagram

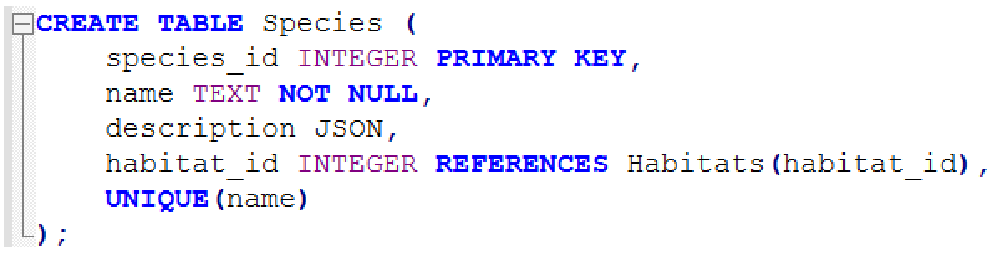


Species:

Columns:

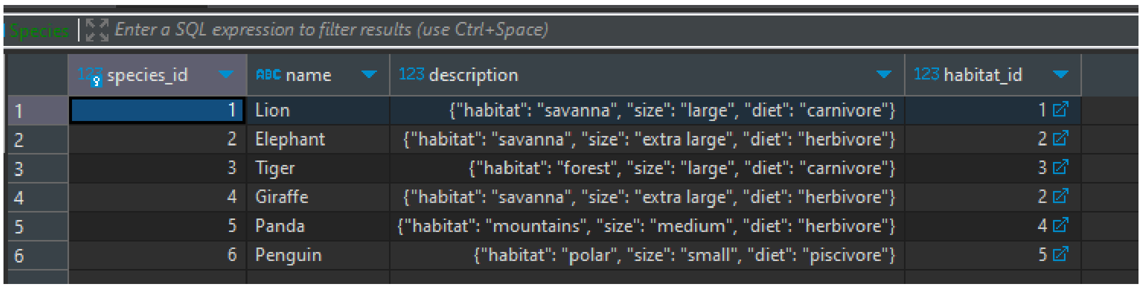
- species_id: Unique identifier for the species (INTEGER, PRIMARY KEY).
- name: Name of the species (TEXT, cannot be null).
- description: Description of the species stored as JSON.
- habitat_id: ID of the habitat where the species resides, foreign key referencing the Habitats table (INTEGER).
- UNIQUE(name): Ensures that the species name is unique.

Definition:



```sql
CREATE TABLE Species (
    species_id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    description JSON,
    habitat_id INTEGER REFERENCES Habitats(habitat_id),
    UNIQUE(name)
);
```

Example:



Species | Enter a SQL expression to filter results (use Ctrl+Space)

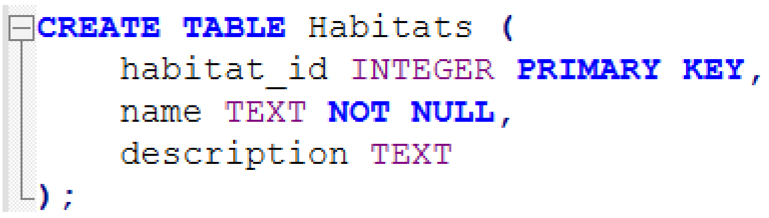
	species_id	name	description	habitat_id
1	1	Lion	{"habitat": "savanna", "size": "large", "diet": "carnivore"}	1
2	2	Elephant	{"habitat": "savanna", "size": "extra large", "diet": "herbivore"}	2
3	3	Tiger	{"habitat": "forest", "size": "large", "diet": "carnivore"}	3
4	4	Giraffe	{"habitat": "savanna", "size": "extra large", "diet": "herbivore"}	2
5	5	Panda	{"habitat": "mountains", "size": "medium", "diet": "herbivore"}	4
6	6	Penguin	{"habitat": "polar", "size": "small", "diet": "piscivore"}	5

Habitats:

Columns:

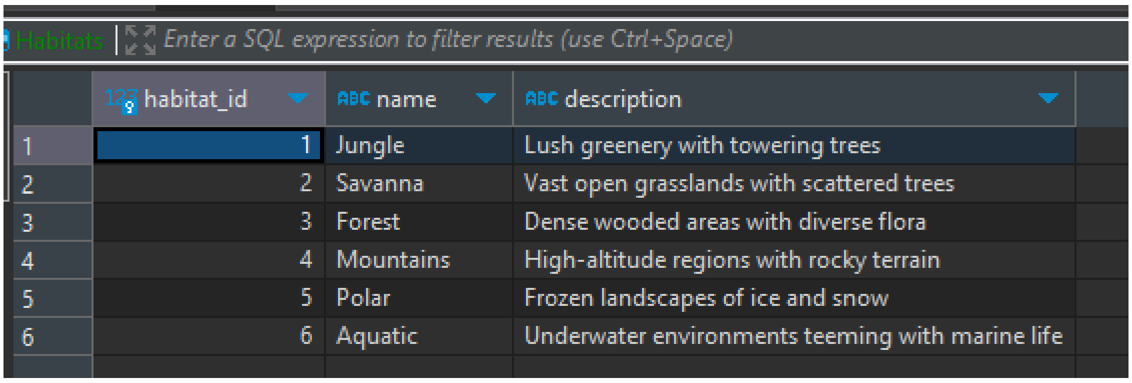
- habitat_id: Unique identifier for the habitat (INTEGER, PRIMARY KEY).
- name: Name of the habitat (TEXT, cannot be null).
- description: Description of the habitat (TEXT).

Definition:



```
CREATE TABLE Habitats (
    habitat_id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    description TEXT
);
```

Example:



Habitats | Enter a SQL expression to filter results (use Ctrl+Space)

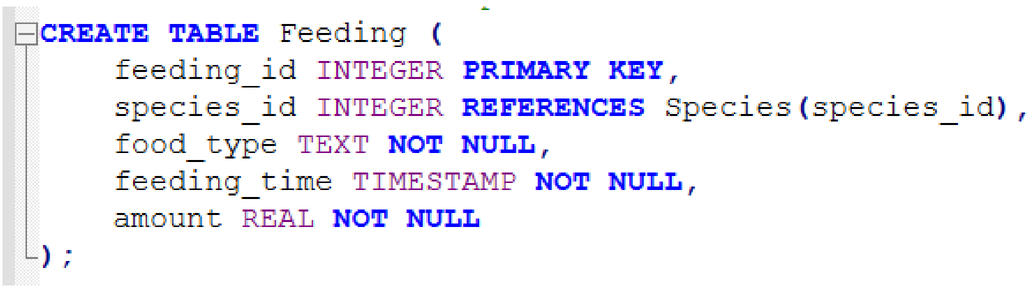
	habitat_id	name	description
1	1	Jungle	Lush greenery with towering trees
2	2	Savanna	Vast open grasslands with scattered trees
3	3	Forest	Dense wooded areas with diverse flora
4	4	Mountains	High-altitude regions with rocky terrain
5	5	Polar	Frozen landscapes of ice and snow
6	6	Aquatic	Underwater environments teeming with marine life

Feeding:

Columns:

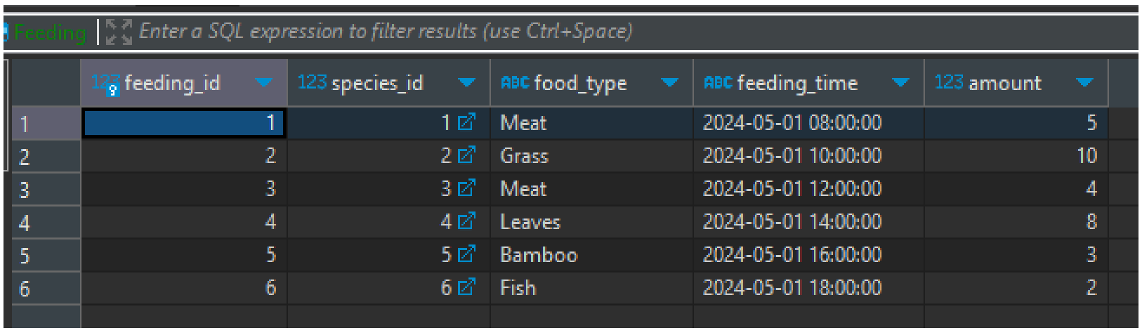
- feeding_id: Unique identifier for the feeding record (INTEGER, PRIMARY KEY).
- species_id: ID of the species being fed, foreign key referencing the Species table (INTEGER).
- food_type: Type of food being fed (TEXT, cannot be null).
- feeding_time: Time of feeding (TIMESTAMP, cannot be null).
- amount: Amount of food given (REAL, cannot be null).

Definition:



```sql
CREATE TABLE Feeding (
    feeding_id INTEGER PRIMARY KEY,
    species_id INTEGER REFERENCES Species(species_id),
    food_type TEXT NOT NULL,
    feeding_time TIMESTAMP NOT NULL,
    amount REAL NOT NULL
);
```

Example:



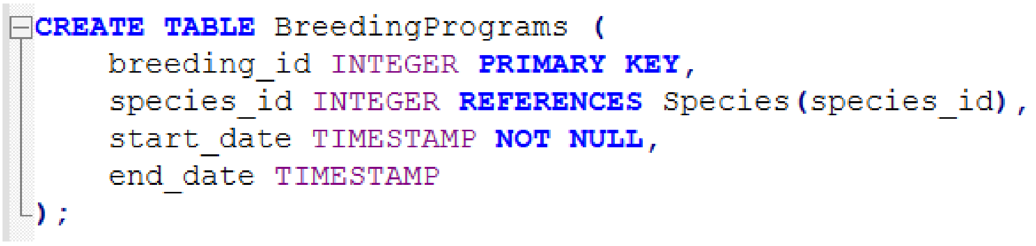
	feeding_id	species_id	food_type	feeding_time	amount
1	1	1	Meat	2024-05-01 08:00:00	5
2	2	2	Grass	2024-05-01 10:00:00	10
3	3	3	Meat	2024-05-01 12:00:00	4
4	4	4	Leaves	2024-05-01 14:00:00	8
5	5	5	Bamboo	2024-05-01 16:00:00	3
6	6	6	Fish	2024-05-01 18:00:00	2

BreedingPrograms:

Columns:

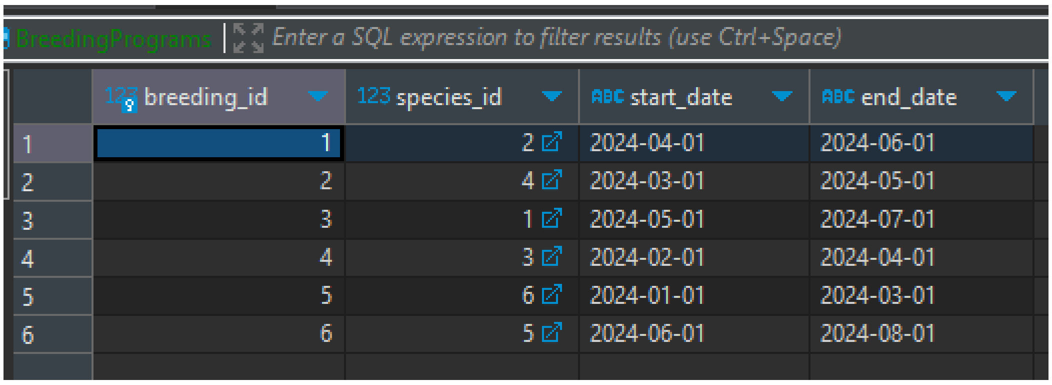
- breeding_id: Unique identifier for the breeding program (INTEGER, PRIMARY KEY).
- species_id: ID of the species involved in the breeding program, foreign key referencing the Species table (INTEGER).
- start_date: Start date of the breeding program (TIMESTAMP, cannot be null).
- end_date: End date of the breeding program (TIMESTAMP).

Definition:



```
CREATE TABLE BreedingPrograms (
    breeding_id INTEGER PRIMARY KEY,
    species_id INTEGER REFERENCES Species(species_id),
    start_date TIMESTAMP NOT NULL,
    end_date TIMESTAMP
);
```

Example:



BreedingPrograms | Enter a SQL expression to filter results (use Ctrl+Space)

	breeding_id	species_id	start_date	end_date
1	1	2	2024-04-01	2024-06-01
2	2	4	2024-03-01	2024-05-01
3	3	1	2024-05-01	2024-07-01
4	4	3	2024-02-01	2024-04-01
5	5	6	2024-01-01	2024-03-01
6	6	5	2024-06-01	2024-08-01

BirthRecords:

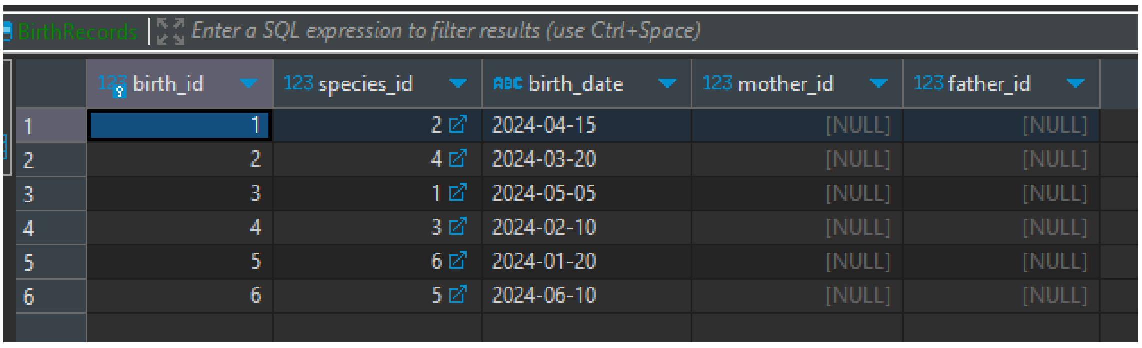
Columns:

- birth_id: Unique identifier for the birth record (INTEGER, PRIMARY KEY).
- species_id: ID of the species born, foreign key referencing the Species table (INTEGER).
- birth_date: Date of birth (TIMESTAMP, cannot be null).
- mother_id: ID of the mother species (INTEGER).
- father_id: ID of the father species (INTEGER).

Definition:

```sql
CREATE TABLE BirthRecords (
    birth_id INTEGER PRIMARY KEY,
    species_id INTEGER REFERENCES Species(species_id),
    birth_date TIMESTAMP NOT NULL,
    mother_id INTEGER,
    father_id INTEGER
);
```

Example:



	birth_id	species_id	birth_date	mother_id	father_id
1	1	2	2024-04-15	[NULL]	[NULL]
2	2	4	2024-03-20	[NULL]	[NULL]
3	3	1	2024-05-05	[NULL]	[NULL]
4	4	3	2024-02-10	[NULL]	[NULL]
5	5	6	2024-01-20	[NULL]	[NULL]
6	6	5	2024-06-10	[NULL]	[NULL]

Visits:

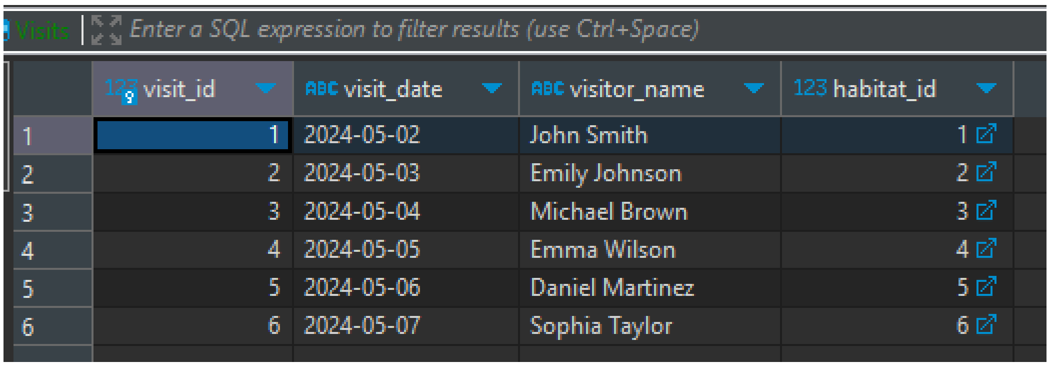
Columns:

- visit_id: Unique identifier for the visit record (INTEGER, PRIMARY KEY).
- visit_date: Date of the visit (TIMESTAMP, cannot be null).
- visitor_name: Name of the visitor (TEXT).
- habitat_id: ID of the habitat visited, foreign key referencing the Habitats table (INTEGER).

Definition:

```sql
CREATE TABLE Visits (
    visit_id INTEGER PRIMARY KEY,
    visit_date TIMESTAMP NOT NULL,
    visitor_name TEXT,
    habitat_id INTEGER REFERENCES Habitats(habitat_id)
);
```

Example:



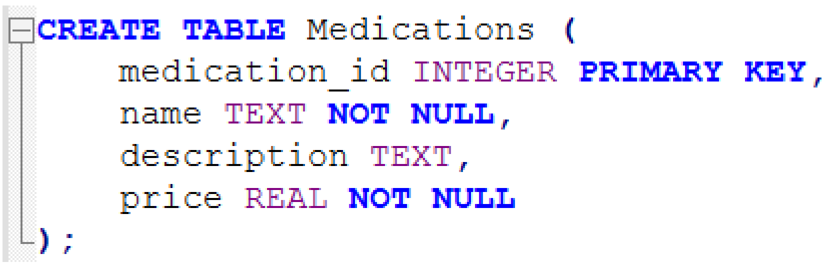
	visit_id	visit_date	visitor_name	habitat_id
1	1	2024-05-02	John Smith	1
2	2	2024-05-03	Emily Johnson	2
3	3	2024-05-04	Michael Brown	3
4	4	2024-05-05	Emma Wilson	4
5	5	2024-05-06	Daniel Martinez	5
6	6	2024-05-07	Sophia Taylor	6

Medications:

Columns:

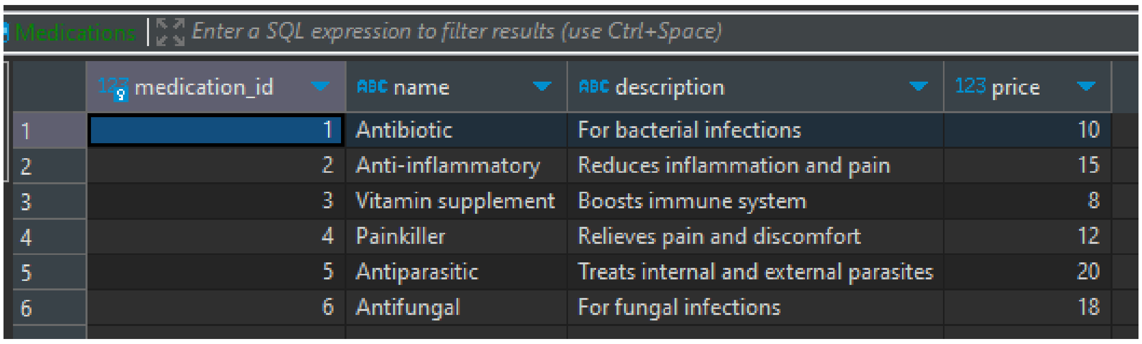
- medication_id: Unique identifier for the medication (INTEGER, PRIMARY KEY).
- name: Name of the medication (TEXT, cannot be null).
- description: Description of the medication (TEXT).
- price: Price of the medication (REAL, cannot be null).

Definition:



```sql
CREATE TABLE Medications (
    medication_id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    description TEXT,
    price REAL NOT NULL
);
```

Example:



Medications | Enter a SQL expression to filter results (use Ctrl+Space)

	medication_id	name	description	price
1	1	Antibiotic	For bacterial infections	10
2	2	Anti-inflammatory	Reduces inflammation and pain	15
3	3	Vitamin supplement	Boosts immune system	8
4	4	Painkiller	Relieves pain and discomfort	12
5	5	Antiparasitic	Treats internal and external parasites	20
6	6	Antifungal	For fungal infections	18

Treatments:

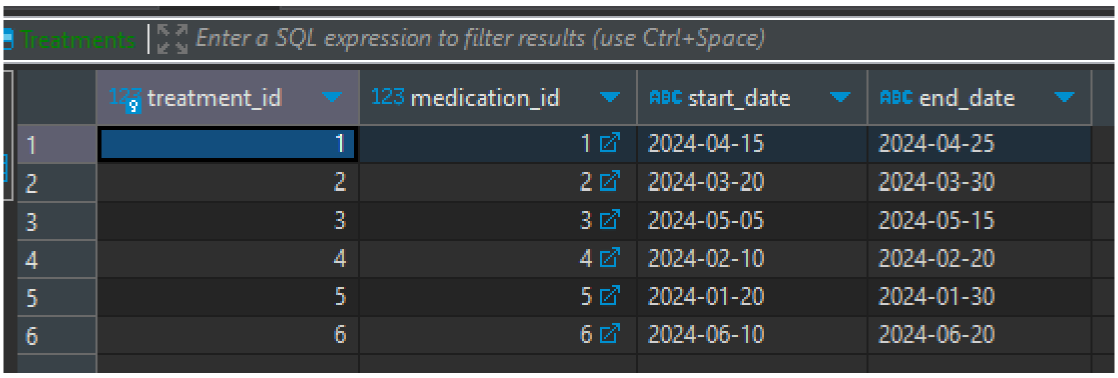
Columns:

- treatment_id: Unique identifier for the treatment (INTEGER, PRIMARY KEY).
- medication_id: ID of the medication used in the treatment, foreign key referencing the Medications table (INTEGER).
- start_date: Start date of the treatment (TIMESTAMP, cannot be null).
- end_date: End date of the treatment (TIMESTAMP).

Definition:

```
CREATE TABLE Treatments (
    treatment_id INTEGER PRIMARY KEY,
    medication_id INTEGER REFERENCES Medications(medication_id),
    start_date TIMESTAMP NOT NULL,
    end_date TIMESTAMP
);
```

Example:



	treatment_id	medication_id	start_date	end_date
1	1	1	2024-04-15	2024-04-25
2	2	2	2024-03-20	2024-03-30
3	3	3	2024-05-05	2024-05-15
4	4	4	2024-02-10	2024-02-20
5	5	5	2024-01-20	2024-01-30
6	6	6	2024-06-10	2024-06-20

AnimalTreatments:

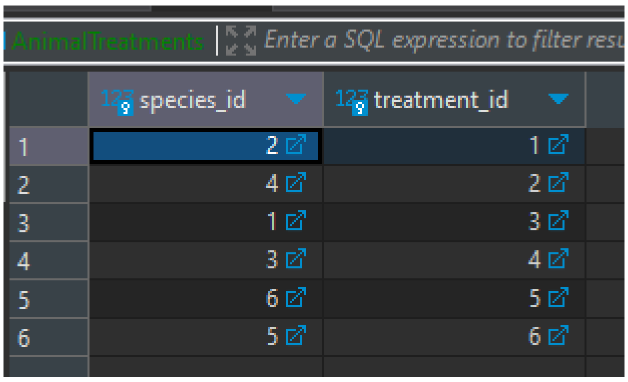
Columns:

- species_id: ID of the species receiving the treatment, part of the composite primary key, foreign key referencing the Species table (INTEGER).
- treatment_id: ID of the treatment administered, part of the composite primary key, foreign key referencing the Treatments table (INTEGER).
- PRIMARY KEY (species_id, treatment_id): Ensures uniqueness of combinations of species and treatments.

Definition:

```sql
CREATE TABLE AnimalTreatments (
    species_id INTEGER REFERENCES Species(species_id),
    treatment_id INTEGER REFERENCES Treatments(treatment_id),
    PRIMARY KEY (species_id, treatment_id)
);
```

Example:


AnimalTreatments | Enter a SQL expression to filter resu

	species_id	treatment_id
1	2	1
2	4	2
3	1	3
4	3	4
5	6	5
6	5	6

Query simples:

-- Get a list of all animal species along with their corresponding habitat.

```
SELECT s.name AS species_name, h.name AS habitat_name
FROM Species s
JOIN Habitats h ON s.habitat_id = h.habitat_id;
```

-- Calculate the average amount of food provided to all species during a specific day.

```
SELECT DATE(feeding_time) AS feeding_date, AVG(amount) AS average_amount
FROM Feeding
GROUP BY feeding_date;
```

-- Find all species that have received a specific treatment.

```
SELECT DISTINCT s.name AS species_name
FROM Species s
JOIN AnimalTreatments at ON s.species_id = at.species_id
JOIN Treatments t ON at.treatment_id = t.treatment_id
WHERE t.medication_id = 1;
```

-- query that uses a Common Table Expression (CTE) to calculate the total treatments administered for each animal species:

```
WITH TreatmentCounts AS (
  SELECT
    at.species_id,
    COUNT(*) AS total_treatments
  FROM
    AnimalTreatments at
  GROUP BY
    at.species_id
)
SELECT
  s.name AS species_name,
  COALESCE(tc.total_treatments, 0) AS total_treatments
FROM
  Species s
LEFT JOIN
  TreatmentCounts tc ON s.species_id = tc.species_id
ORDER BY
  total_treatments DESC;
```