Advanced SQL
Forum: `https://forum-db.informatik.uni-tuebingen.de/c/ss20-asql`

Assignment 4

Relevant videos: up to #24
`https://tinyurl.com/AdvSQL-2020`
Submission: Tuesday, 26.05.2020, 10:00 AM

1. [10 Points] **What type are you?**

   Consider the following SQL query with embedded values:

```
SELECT p.a, p.b * 2, p.c, p.d, p.e, p.f
FROM   (VALUES
  (1,'2'::money,4    ,41+1::real,1::real,NULL),
  (2,'5.72'     ,1.32,2          ,2       ,NULL),
  (3,'2'::money,5.77,3          ,3       ,NULL)
) AS p(a,b,c,d,e,f)
WHERE  p.c < 5.5;
```

   Extract the `FROM`-clause of this query into a permanent table `P` using the DDL (i.e.: `CREATE TABLE`) and DML (i.e.: `INSERT`). Then, run the following query derived from the query above. It must yield the same **exact** results.

```
SELECT p.a, p.b * 2, p.c, p.d, p.e, p.f
FROM   p AS p
WHERE  p.c < 5.5;
```

   Hint: Types can be detemined in PostgreSQL by using `pg_typeof(...)`[1].

2. [20 Points] **Magic the JSONing**

   We provided you with a compressed JSON file `AllCards.json.zip` which encodes a list of cards for a popular collectible card game. The format of this JSON file is well documented in `mtj-doc.html`.

   (a) To load the data from the JSON file into your database, follow these steps: First, extract `AllCards.json` from `AllCards.json.zip`. Then, load the JSON file into a **temporary** table `allcards_json` defined as follows:

```
CREATE TABLE allcards_json (
  data json
);

\copy allcards_json FROM 'path/to/AllCards.json/here';
```

   Next, define a permanent table `mtj`:

```
CREATE TABLE mtj (
  name      text PRIMARY KEY,
  mana_cost text,
  cmc       numeric,
  type      text,
  text      text,
  power     text,
  toughness text
);
```

[1] `https://www.postgresql.org/docs/12/functions-info.html#FUNCTIONS-INFO-CATALOG-TABLE`

Finally, write a SQL query which uses the JSON format of table `allcards_json` to populate table `mtj` with its respective values.
After completing these steps, write SQL queries using the now populated table `mtj`:

(b) List the cards with minimum and maximum `cmc` value. Disregard cards with a `cmc` value of 0 or less.

(c) List how many cards exist with `type Enchantment` excluding cards of type `Enchantment-Aura`.

(d) List the first five cards with the highest `cmc`. The listed cards should satisfy the following predicates: `Power` or `toughness` are greater than 14 or `power` is less than `toughness`. Ignore cards with `power` or `toughness` containing the character `*`.

(e) List how many cards exist with `mana_cost` of **exactly** one, two or three {`U`}.

(f) For **this task only** write two SQL queries. One using `mtj` and another accessing the JSON data structure in `allcards_json` directly. List the names of all cards where the text contains `Recover` while having a `CMC` of 2 or lower. For both query variants, the result will be a table whose single cell holds a JSON array containing JSON objects. Each JSON object has exactly one field `name` where the name of the card is recorded:

```
[{"name":"Card1"}, {"name":"Card2"}, ...]
```

Note: Make use of the many built-in JSON functions[2] to keep your queries simple.

[2] `https://www.postgresql.org/docs/12/functions-json.html`