EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Advanced SQL
Sommersemester 2022
Prof. Torsten Grust, Christian Duta
WSI — Database Systems Research Group

Assignment 4 (20.05.2022)

Handin until: 27.05.2022, 09:00

1. [4 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 values in the `FROM`-clause 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: In PostgreSQL, the type of an expression `e` can be detemined by `pg_typeof(e)`[1].

2. [13 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 table `allcards_json` defined as follows:

```
CREATE TABLE allcards_json (
  data jsonb
);

\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
);
```

   Finally, write a SQL query that 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`:

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

(b) List the names of the top five cards with the highest `cmc` which satisfy the following predicates: `Power` or `toughness` are greater than 14 or `power` is less than `toughness`. Disregard cards with `power` or `toughness` containing the character `*`.
Hint: Use function `translate`[2] to remove any trailing and leading `"` from `power` and `toughness` to cast these values to `float`.

(c) Count how many cards exist with `mana_cost` of **exactly** `{U}`, `{U}{U}` or `{U}{U}{U}`.

(d) For **this task only** write two SQL queries. One using table `mtj` and another accessing the JSON data structure in `allcards_json` directly. List the names of all cards where the text contains `Recover` that have a `CMC` of 2 or less.
The result of both queries is a table with a single cell that holds a JSON array containing the card names as JSON objects. Each JSON object has exactly one field `name` where the name of the card is recorded:

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

Then, compare the evaluation times of both queries. **Explain your observations.**
Note: Make use of the many built-in JSON functions[3] to keep your queries simple. Ensure that you are generating values of type `jsonb` (not `json`).

3. [13 Points] **Earthquakes**

We provided you with with a file `earthquakes.zip`. It contains two files:

- `earthquakes.json` encodes a list of earthquakes detected between 17. April and 17. May 2022. The format of this JSON file is well documented[4][5].
- `earthquakes.sql` copies the JSON data from `earthquakes.json` into a table `earthquakes`:

```
CREATE TABLE earthquakes (
  title text,
  quake jsonb
);
```

Run `earthquakes.sql` and then write the following queries which use table `earthquakes`:

(a) Find the `title` and `depth` of those earthquakes with the highest and lowest `depth`.

(b) List the `title` and `date` value of all earthquakes with the highest `magnitude` on each day in the dataset.
Hint: Use `to_timestamp(time / 1000)` to convert the `time` (in milliseconds since `1970-01-01T00:00:00.000Z`) to a timestamp (which can then be casted into a `date` value).

(c) Find the `title` of the earthquake that occured closest to the Sand (longitude: 48.534542, latitude: 9.071296). To calculate the distance between two points p_1, p_2, use the *haversine* formula:

$$d = 2r \cdot arcsin\left(\sqrt{sin^2(\frac{radians(\phi_2-\phi_1)}{2}) + cos(radians(\phi_1)) \cdot cos(radians(\phi_2)) \cdot sin^2(\frac{radians(\lambda_2-\lambda_1)}{2})}\right)$$, where

- d is the distance between p_1 and p_2
- r is the radius of the sphere (here: Earth's radius = $6371km$)
- ϕ_1, ϕ_2 is the latitude of p_1 and p_2, respectively
- λ_1, λ_2 is the longitude of p_1 and p_2, respectively
- *radians* converts degrees to radians

We provided you with the *haversine* function in form of a *SQL* function

```
haversine(lat_p1 float, lon_p1 float, lat_p2 float, lon_p2 float)
```

in `earthquakes.sql`.

[2] https://www.postgresql.org/docs/current/functions-string.html#FUNCTIONS-STRING-OTHER
[3] https://www.postgresql.org/docs/current/functions-json.html
[4] https://earthquake.usgs.gov/earthquakes/feed/v1.0/geojson.php
[5] https://earthquake.usgs.gov/data/comcat/data-eventterms.php