11

Faça como eu fiz

1) Para esta aula faça o download do arquivo **ExportEmpresa.sql**. Execute este script para criar uma nova base de dados no MySQL.

2) Para verificar o campo JSON que iremos trabalhar execute:

```
SELECT * FROM tb_object_funcionario;
SELECT JSON_PRETTY(JSON) FROM tb_object_funcionario;
```

3) Vamos extrair algumas propriedades da tabela **tb_object_funcionario** que possui um campo do tipo JSON.

```
SELECT
JSON_EXTRACT(JSON,"$.Primeiro_Nome")
,JSON_EXTRACT(JSON,"$.Data_Nascimento")
,JSON_EXTRACT(JSON,"$.Salario")
FROM tb_object_funcionario;
```

4) Em aula passada vimos o uso do comando REPLACE para retirar as aspas duplas do resultado do campo que é extraído do JSON.

```
SELECT
REPLACE(JSON_EXTRACT(JSON,"$.Primeiro_Nome"),'"','')
,JSON_EXTRACT(JSON,"$.Data_Nascimento")
,JSON_EXTRACT(JSON,"$.Salario")
FROM tb_object_funcionario;
```

5) Porém, há um comando que pode ser usado para substituir o uso do REPLACE.

```
SELECT
JSON_UNQUOTE(JSON_EXTRACT(JSON,"$.Primeiro_Nome"))
,JSON_UNQUOTE(JSON_EXTRACT(JSON,"$.Data_Nascimento"))
,JSON_EXTRACT(JSON,"$.Salario")
FROM tb_object_funcionario;
```

6) Vamos verificar o tipo de propriedade que o JSON da tabela **tb_object_funcionario** possui.

```
SELECT
JSON_TYPE(JSON_EXTRACT(JSON,"$.Primeiro_Nome"))
,JSON_TYPE(JSON_EXTRACT(JSON,"$.Data_Nascimento"))
,JSON_TYPE(JSON_EXTRACT(JSON,"$.Salario"))
FROM tb_object_funcionario;
```

7) Apesar dos tipos não corresponderem, o MySQL consegue identificar o campo que tem formato de data e permite o uso de funções específicas para trabalha com este tipo de campo. Veja o comando abaixo:

```
SELECT
JSON_UNQUOTE(JSON_EXTRACT(JSON,"$.Primeiro_Nome"))
,JSON_UNQUOTE(JSON_EXTRACT(JSON,"$.Data_Nascimento"))
,JSON_EXTRACT(JSON,"$.Salario")
FROM tb_object_funcionario
WHERE YEAR(JSON_EXTRACT(JSON,"$.Data_Nascimento")) >= 1980
```

8) Em aula anterior trabalhamos com arrays. Mas, nos nossos exemplos, os arrays possuíam elementos de um único tipo. Agora vamos trabalhar com array de JSONs. Aqui cada elemento de um array é um JSON. Inicialmente vamos revisar como retiramos dados de um array simples.

```
SELECT '["PRAIA","FUTEBOL","CINEMA"]';
SELECT JSON_EXTRACT('["PRAIA","FUTEBOL","CINEMA"]',"$[1]");
SELECT '{"HOBBY":["PRAIA","FUTEBOL","CINEMA"]}';
SELECT JSON_EXTRACT('{"HOBBY":["PRAIA","FUTEBOL","CINEMA"]}',"$.HOBBY[1]");
```

9) Agora veja a diferença quando temos um array de JSONs:

```
SELECT JSON_EXTRACT('{"HOBBY":[{"nome":"PRAIA","local":"Ar Livre"},{"nome":"FUTEBOL","local":"A
SELECT JSON_EXTRACT('{"HOBBY":[{"nome":"PRAIA","local":"Ar Livre"},{"nome":"FUTEBOL","local":"A
SELECT JSON_EXTRACT('{"HOBBY":[{"nome":"PRAIA","local":"Ar Livre"},{"nome":"FUTEBOL","local":"A
{"nome":"CINEMA","local":"Fechado"}]}','$.HOBBY[0].nome')
UNION
SELECT JSON_EXTRACT('{"HOBBY":[{"nome":"PRAIA","local":"Ar Livre"},{"nome":"FUTEBOL","local":"A
{"nome":"CINEMA","local":"Fechado"}]}','$.HOBBY[1].nome')
UNION
SELECT JSON_EXTRACT('{"HOBBY":[{"nome":"PRAIA","local":"Ar Livre"},{"nome":"FUTEBOL","local":"A
{"nome":"CINEMA","local":"Fechado"}]}','$.HOBBY[2].nome');
```

10) Muitas vezes o array de JSONs precisam que seus dados sejam extraídos para formar uma tabela sequencial, onde em cada linha temos um elemento do array. Podemos fazer isso usando diversos UNIONs como mostrado abaixo:

```
CREATE TABLE X (Y JSON);
INSERT INTO X VALUES ('{"HOBBY":[{"nome":"PRAIA","local":"Ar Livre"},{"nome":"FUTEBOL","local":'
{"nome":"CINEMA","local":"Fechado"}]}');
SELECT Y FROM X;

SELECT JSON_EXTRACT(Y,"$.HOBBY[0].nome") AS NOME, JSON_EXTRACT(Y,"$.HOBBY[0].local") AS LOCAL FI
UNION
SELECT JSON_EXTRACT(Y,"$.HOBBY[1].nome") AS NOME, JSON_EXTRACT(Y,"$.HOBBY[1].local") AS LOCAL FI
UNION
SELECT JSON_EXTRACT(Y,"$.HOBBY[2].nome") AS NOME, JSON_EXTRACT(Y,"$.HOBBY[2].local") AS LOCAL FI
```

11) Isso nos apresenta um problema. Toda a vez que um novo elemento do array de JSONs é incluído nosso comando SQL, que possui UNIONs, deve ter mais uma sentença. Na verdade, terei que ter 1 UNION para cada elemento do array.

```
UPDATE X SET Y = '{"HOBBY":[{"nome":"PRAIA","local":"Ar Livre"},{"nome":"FUTEBOL","local":"Ar Li
{"nome":"CINEMA","local":"Fechado"}, {"nome":"PISCINA","local":"Ar Livre"}]}';
```

```
SELECT JSON_EXTRACT(Y,"$.HOBBY[0].nome") AS NOME, JSON_EXTRACT(Y,"$.HOBBY[0].local") AS LOCAL FI
UNION
SELECT JSON_EXTRACT(Y,"$.HOBBY[1].nome") AS NOME, JSON_EXTRACT(Y,"$.HOBBY[1].local") AS LOCAL FI
UNION
SELECT JSON_EXTRACT(Y,"$.HOBBY[2].nome") AS NOME, JSON_EXTRACT(Y,"$.HOBBY[2].local") AS LOCAL FI
UNION
SELECT JSON_EXTRACT(Y,"$.HOBBY[3].nome") AS NOME, JSON_EXTRACT(Y,"$.HOBBY[3].local") AS LOCAL FI
```

12) Com o uso do JSON_TABLE este problema é solucionado com uma única consulta que irá ser independente do número de elementos do array de JSONs.

```
SELECT T2.NOME, T2.LOCAL FROM X
CROSS JOIN
JSON_TABLE ( JSON_EXTRACT (Y, "$.HOBBY"), "$[*]"
COLUMNS (NOME VARCHAR(10) PATH "$.nome", LOCAL VARCHAR(10) PATH "$.local")) T2;

UPDATE X SET Y = '{"HOBBY":[{"nome":"PRAIA","local":"Ar Livre"},{"nome":"FUTEBOL","local":"Ar Li
{"nome":"CINEMA","local":"Fechado"}, {"nome":"PISCINA","local":"Ar Livre"}, {"nome":"TENIS","loc
```

13) Vamos aplicar o JSON_TABLE para listar as características dos dependentes dos funcionário usando a base que foi recuperada nesta aula. A solução é mostrada abaixo:

```
SELECT
JSON_UNQUOTE(JSON_EXTRACT(JSON, "$.Cpf")) AS CPF,
JSON_UNQUOTE(JSON_EXTRACT(JSON, "$.Primeiro_Nome")) AS PRIMEIRO_NOME,
JSON_UNQUOTE(JSON_EXTRACT(JSON, "$.Nome_Meio")) AS NOME_MEIO,
JSON_UNQUOTE(JSON_EXTRACT(JSON, "$.Ultimo_Nome")) AS ULTIMO_NOME,
TD.NOME_DEPARTAMENTO as NOME_DO_DEPARTAMENTO,
T1.NOME_DEPENDENTE, T1.SEXO, T1.PARENTESCO, T1.DATA_NASCIMENTO FROM tb_object_funcionario
CROSS JOIN
JSON_TABLE (JSON_EXTRACT (JSON, "$.Dependentes"), "$[*]"
COLUMNS (NOME_DEPENDENTE VARCHAR(30) PATH "$.Nome_Dependente",
SEXO VARCHAR(2) PATH "$.Sexo",
PARENTESCO VARCHAR(20) PATH "$.Parentesco",
DATA_NASCIMENTO DATETIME PATH "$.Data_Nascimento")) T1
INNER JOIN tb_departamento TD ON
TD.NUMERO_DEPARTAMENTO = JSON_UNQUOTE(JSON_EXTRACT(JSON, "$.Numero_Departamento"))
```