15

Faça como eu fiz

1) Abra o **MySQL Workbench**. Use a conexão já configurada em outros cursos na carreira de MySQL. Se você não possui o MySQL e nem o MySQL instalados volte ao curso **Introdução ao SQL com MySQL** - **Manipule e consulte os dados** , na Aula1, e instale os produtos.

2) Usando um editor de JSON online crie a seguinte expressão:

```
{"name":"Bond","first":"James","ID":"007"}
```

3) Volte ao **MySQL Workbench** e crie um banco de dados chamado **json_curso**.

4) Crie uma tabela executando o comando:

```
CREATE TABLE foo (oldjson char(250));
```

5) Inclua o JSON na tabela criada:

```
INSERT INTO foo VALUES ('{"name":"Bond","first":"James","ID":"007"}');
```

6) O JSON foi incluído num campo do tipo **VARCHAR**. Fazendo isso o **MySQL** não vai entender que est campo é do tipo JSON. Logo temos que criar a tabela de uma forma diferente, inserindo o mesmo valor:

```
CREATE TABLE bar (our_data JSON);
INSERT INTO bar VALUES ('{"name":"Bond","first":"James","ID":"007"}');
```

7) Agora sim o campo criado na tabela será do tipo JSON. Insira nova linha:

```
INSERT INTO bar VALUES ('{"name":"Smart","first":"Maxwell","ID":"86"}');
```

8) Se executamos a consulta abaixo o JSON é exibido, mas de forma desorganizada:

```
SELECT * FROM bar;
```

9) Mas, se executarmos usando a função apropriada o JSON será criado de forma correta.

```
SELECT JSON_PRETTY(our_data) FROM bar;
```

10) Faça o download do arquivo **world_x_v2.sql**

11) Execute este script no **MySQL Workbench** para criar a base de estudos.

12) Execute a consulta abaixo para entender o campo JSON que será usado nos próximos testes.

```
SELECT DOC FROM countryinfo WHERE _id = 'USA';
```

13) A função abaixo mostra o nome das propriedades existentes no campo JSON:

```
SELECT JSON_KEYS(doc) FROM countryinfo WHERE _id = 'USA';
```

14) Podemos extrair o valor de uma propriedade do campo JSON:

```
SELECT JSON_KEYS(doc, "$.geography") FROM countryinfo WHERE _id = 'USA';
```

15) Ou então um valor de uma propriedade dentro de outra propriedade:

```
SELECT JSON_EXTRACT(doc, "$.government.HeadOfState") FROM countryinfo WHERE _id = 'USA';
```

16) Execute o SQL abaixo para ver todas as propriedades do JSON:

```
SELECT JSON_EXTRACT(doc, "$.GNP") as GNP
  , JSON_EXTRACT(doc, "$.Code") as Code
  , JSON_EXTRACT(doc, "$.Name") as Name
  , JSON_EXTRACT(doc, "$.IndepYear") as IndepYear
  , JSON_EXTRACT(doc, "$.geography.Region") as Region
  , JSON_EXTRACT(doc, "$.geography.Continent") as Continent
  , JSON_EXTRACT(doc, "$.geography.SurfaceArea") as SurfaceArea
  , JSON_EXTRACT(doc, "$.government.HeadOfState") as HeadOfState
  , JSON_EXTRACT(doc, "$.government.GovernmentForm") as GovernmentForm
  , JSON_EXTRACT(doc, "$.demographics.Population") as Population
  , JSON_EXTRACT(doc, "$.demographics.LifeExpectancy") as LifeExpectancy
  FROM countryinfo;
```

17) Use a função REPLACE para tirar as aspas duplas do resultado das consultas de extração de valores das propriedades no campo JSON.

```
SELECT JSON_EXTRACT(doc, "$.Code") as Code, REPLACE(JSON_EXTRACT(doc, "$.Code"),'"','') as Code2
FROM countryinfo;
```

18) Os campos extraídos podem fazer parte de uma consulta grupando linhas de uma tabela. Execute:

```
SELECT JSON_EXTRACT(doc, "$.geography.Continent") as Continent,
SUM(JSON_EXTRACT(doc, "$.demographics.Population")) as Population,
AVG(JSON_EXTRACT(doc, "$.demographics.LifeExpectancy")) FROM countryinfo
GROUP BY JSON_EXTRACT(doc, "$.geography.Continent")
ORDER BY 2;
```

19) Também podemos usar os valores extraídos em filtros na tabela.

```
SELECT JSON_EXTRACT(doc, "$.geography.Continent") as Continent,
SUM(JSON_EXTRACT(doc, "$.demographics.Population")) as Population,
AVG(JSON_EXTRACT(doc, "$.demographics.LifeExpectancy")) FROM countryinfo
WHERE JSON_EXTRACT(doc, "$.government.GovernmentForm") LIKE ('%Monarchy%')
AND JSON_EXTRACT(doc, "$.demographics.Population") >= 10000000
GROUP BY JSON_EXTRACT(doc, "$.geography.Continent")
ORDER BY 2;
```

20) Para os próximos testes crie uma nova tabela:

```
CREATE TABLE X (Y JSON);
INSERT INTO X VALUES ('{"nome":"Joao", "telefone":"2293-3343"}');
INSERT INTO X VALUES ('{"nome":"Jonas"}');
```

21) A função abaixo mostra se uma determinada propriedade existe no campo JSON. Execute:

```
SELECT JSON_CONTAINS_PATH(Y, "ONE", "$.telefone") FROM X;
```

22) Faça as seguintes inclusões de novos campos na tabela criada mesclando com consultas usando a função que determina se a propriedade existe no JSON:

```
INSERT INTO X VALUES ('{"nome":"Alberto","endereco":"Rua X numero Y"}');
SELECT JSON_CONTAINS_PATH(Y, "ONE", "$.telefone") FROM X;

SELECT Y FROM X;

SELECT JSON_CONTAINS_PATH(Y, "ONE", "$.telefone", "$.endereco") FROM X;
INSERT INTO X VALUES ('{"nome":"Maria","endereco":"Rua X numero Y", "telefone":"2293-3343"}');
SELECT Y FROM X;
SELECT JSON_CONTAINS_PATH(Y, "ONE", "$.telefone", "$.endereco") FROM X;
SELECT JSON_CONTAINS_PATH(Y, "ALL", "$.telefone", "$.endereco") FROM X;
```

23) Já se quero buscar um valor de uma propriedade usamos a função abaixo:

```
SELECT * FROM X;
SELECT JSON_CONTAINS(Y, '"2293-3343"', "$.telefone") FROM X;
SELECT *  FROM X WHERE JSON_CONTAINS(Y, '"2293-3343"', "$.telefone") = 1;
SELECT *  FROM X WHERE JSON_EXTRACT(Y, "$.telefone") = '2293-3343';
```

24) Já, usando a JSON_SEARCH, achamos o valor em qualquer propriedade:

```
SELECT JSON_SEARCH(Y, "ONE", "2293-3343"), Y FROM X;
INSERT INTO X VALUES ('{"nome":"Katia","endereco":"Rua X numero Y", "telefone":"2293-3343", "te
SELECT * FROM X;

SELECT JSON_SEARCH(Y, "ONE", "2293-3343"), Y FROM X;
SELECT JSON_SEARCH(Y, "ALL", "2293-3343"), Y FROM X;
```