**Normalisation-**

Transaction table

|  |  |  |  |
| --- | --- | --- | --- |
| Order\_id | Store\_location | Customer\_id | Amount |
| 1 | New York | C01 | 1450 |
| 2 | LA | C34 | 1250 |
| 3 | Miami | C12 | 1000 |

Order details

|  |  |  |  |
| --- | --- | --- | --- |
| Order\_id | Product\_id | Product\_name | price |
| 1 | 1234 | Shampoo | 240 |
| 1 | 234 | Grinder | 3400 |
| 2 | 34 | Table | 4700 |

Customer table

|  |  |
| --- | --- |
| Customer\_id | Customer\_name |
| C01 | Pit |
| C12 | Jack |

**Denormalization-**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Order\_id | Store\_location | Customer\_id | Customer\_name | Amount | Product\_id | Product\_name | price |
| **1** | New York | C01 | Pit | 1450 | 1234 | Shampoo | 240 |
| **1** | New York | C01 | Pit | 1450 | 234 | Grinder | 3400 |

**BigQuery Table**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Order\_id | Store\_location | Customer\_id | Customer\_name | Amount | Product\_id | Product\_name | price |
| **1** | New York | C01 | Pit | 1450 | 1234 | Shampoo | 240 |
|  |  |  |  |  | 234 | Grinder | 3400 |
| **2** | LA | C34 | Ryan | 1250 | 34 | Table | 4700 |
|  |  |  |  |  | 23 | Dryer | 235 |

BigQuery supports columns with nested and repeated data.

Since every order has multiple products, therefore array of those products. Also product table will have multiple column of product details, Array will be of structure type.

JSON format

{“order\_id”:”1”,”store\_location”:”New York”,”Amount”:”1450”,”customer\_id”:”C01”,

“products”:**[**{“product\_id”:”1234”,” Product\_name”:”Shampoo”,” price”:”240”},

{“product\_id”:”234”,” Product\_name”:”Grinder”,” price”:”3400”},**]**}

{“order\_id”:”2”,”store\_location”:”LA”,”Amount”:”1250”,”customer\_id”:”C34”,

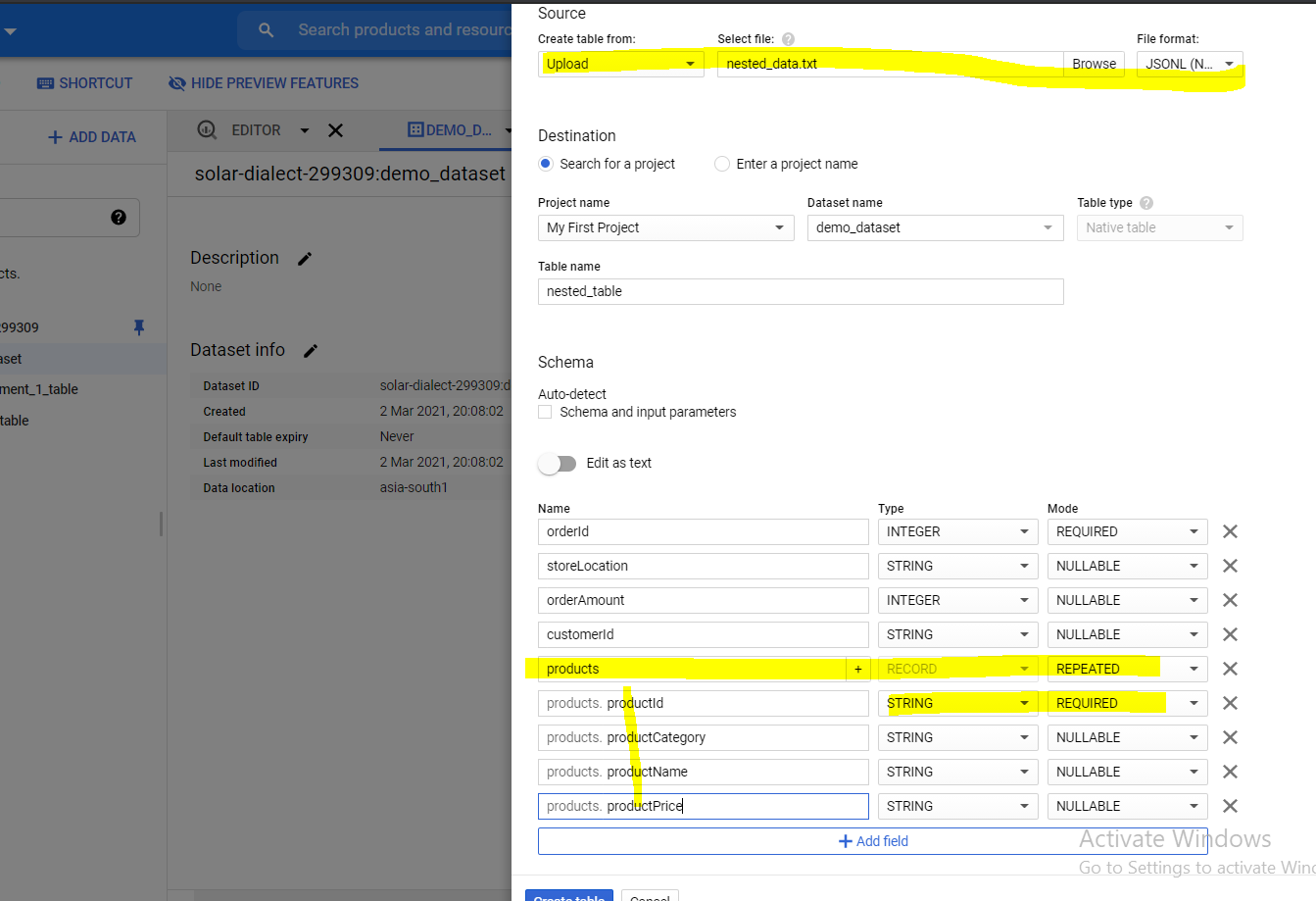
“products”:**[**{“product\_id”:”34”,” Product\_name”:”Table”,” price”:”4700”},

{“product\_id”:”23”,” Product\_name”:”Dryer”,” price”:”235”},**]**}

Details of all products for order 1, has been put in array in products. Each element of that array has structure of product table. (struct type array)

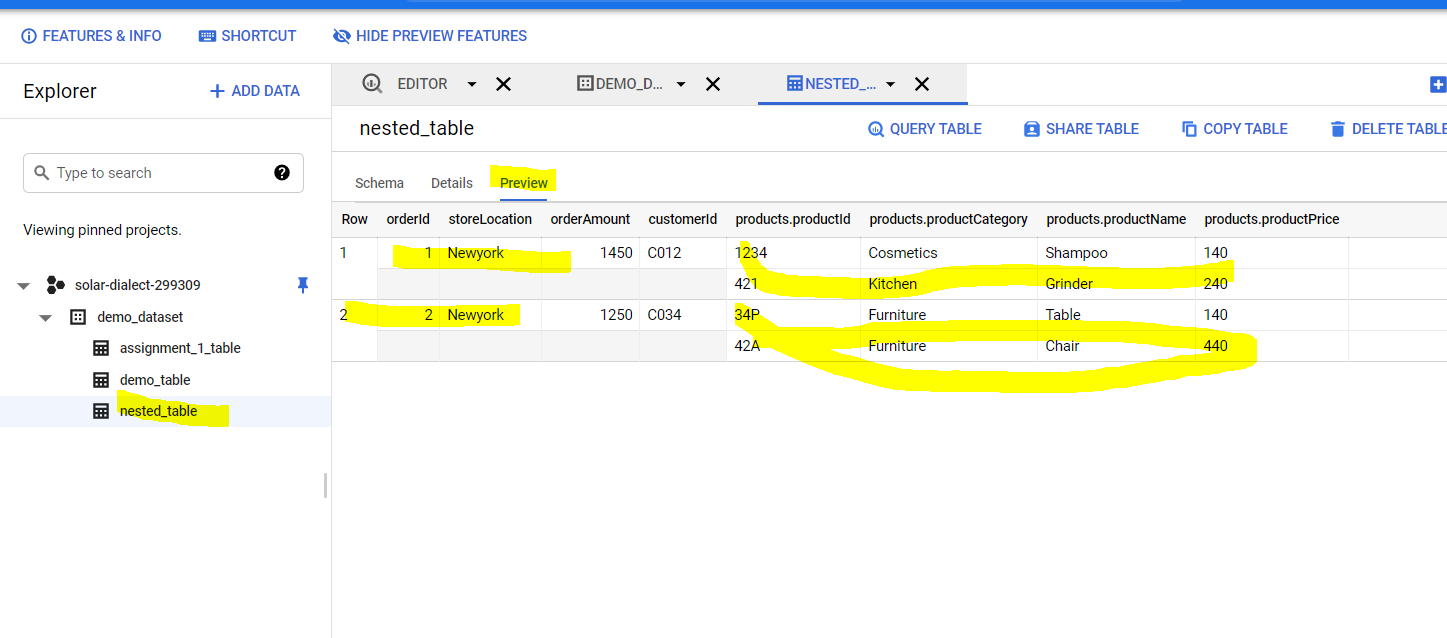
**Nested Table in BQ**





Column name ‘PRODUCTS’, type will be ‘RECORD’ (or array) and it will be repeated.

And we can add sub-columns for PRODUCTS (structure)



**Querying nested table**

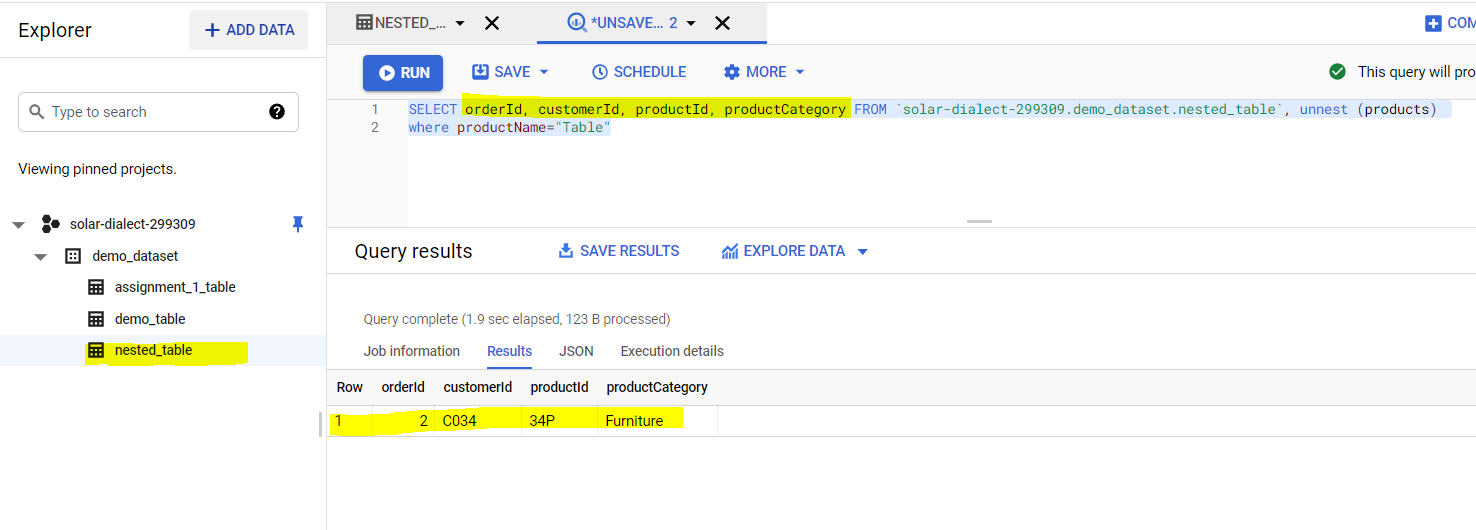
unnext(table\_name)

SELECT \* FROM `solar-dialect-299309.demo\_dataset.nested\_table`, unnest (products)

Cross-joining nested\_table and products table

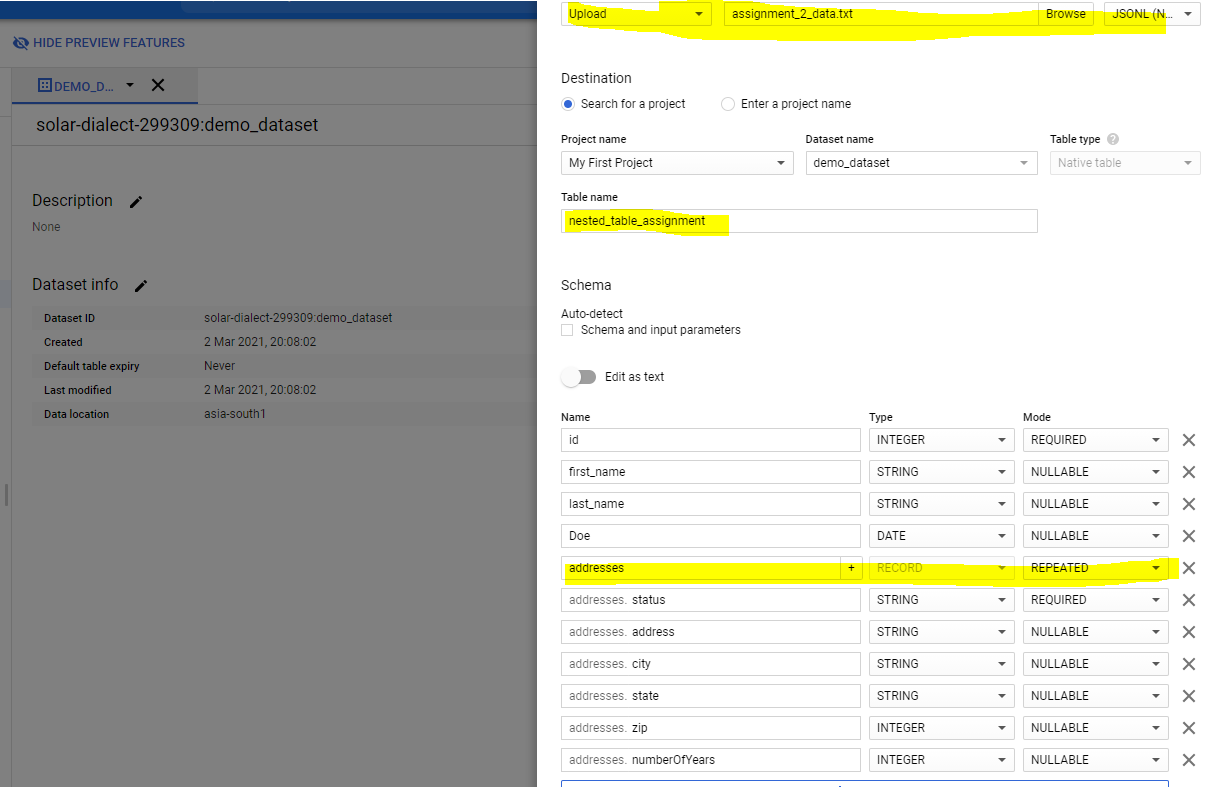
|  |
| --- |
|  |
|  | 1 | Newyork | 1450 | C012 | 1234 | Cosmetics | Shampoo | 140 | 1234 | Cosmetics | Shampoo | 140 |
|  |  |  |  |  | 421 | Kitchen | Grinder | 240 |  |  |  |  |  |
|  | 1 | Newyork | 1450 | C012 | 1234 | Cosmetics | Shampoo | 140 | 421 | Kitchen | Grinder | 240 |  |
|  |  |  |  |  | 421 | Kitchen | Grinder | 240 |  |  |  |  |  |
|  | 2 | Newyork | 1250 | C034 | 34P | Furniture | Table | 140 | 34P | Furniture | Table | 140 |  |
|  |  |  |  |  | 42A | Furniture | Chair | 440 |  |  |  |  |  |
|  | 2 | Newyork | 1250 | C034 | 34P | Furniture | Table | 140 | 42A | Furniture | Chair | 440 |  |
|  |  |  |  |  | 42A | Furniture | Chair | 440 |  |  |  |  |  |

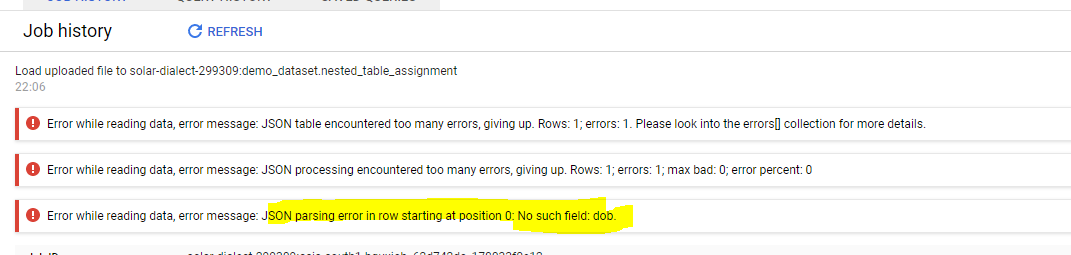
Cross join and ‘where’ condition



**Assignment 2**







(column name is’DOB’, I wrongly mentione ‘DOE’, hence error)

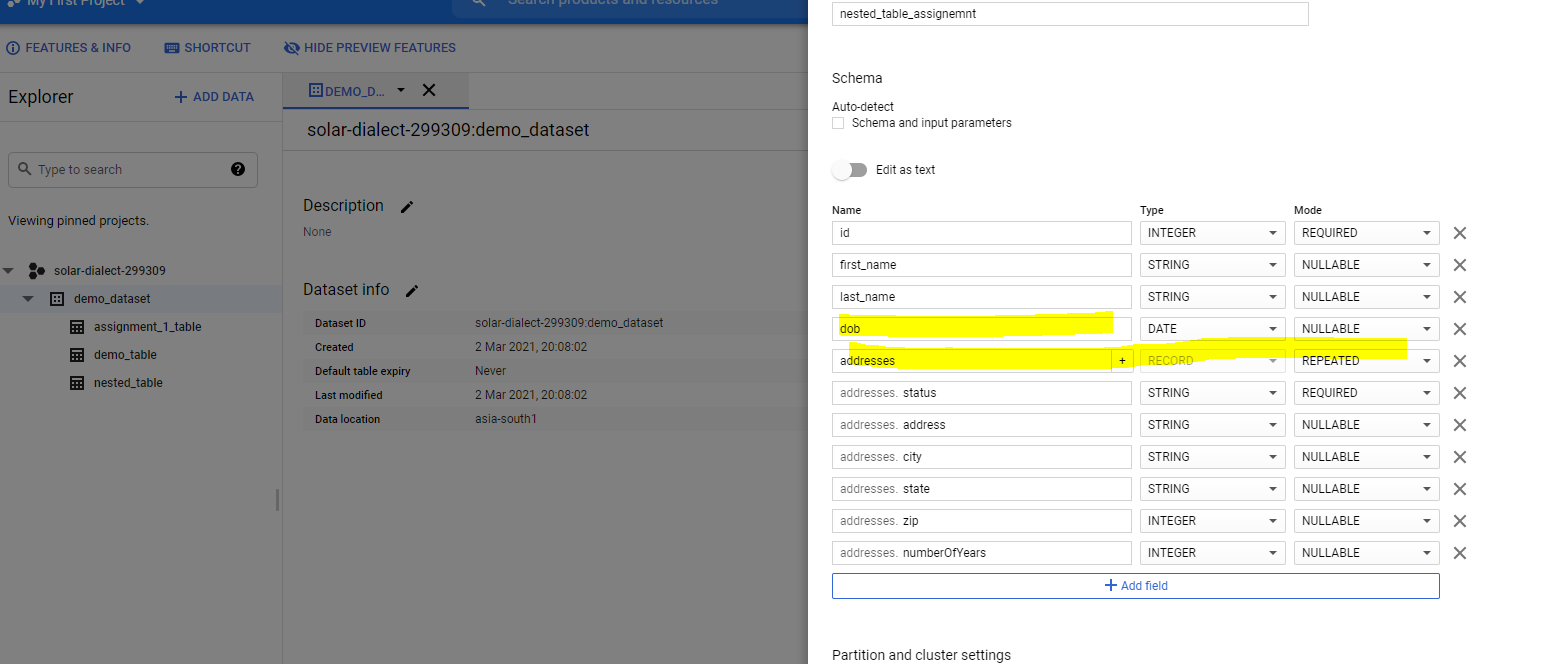
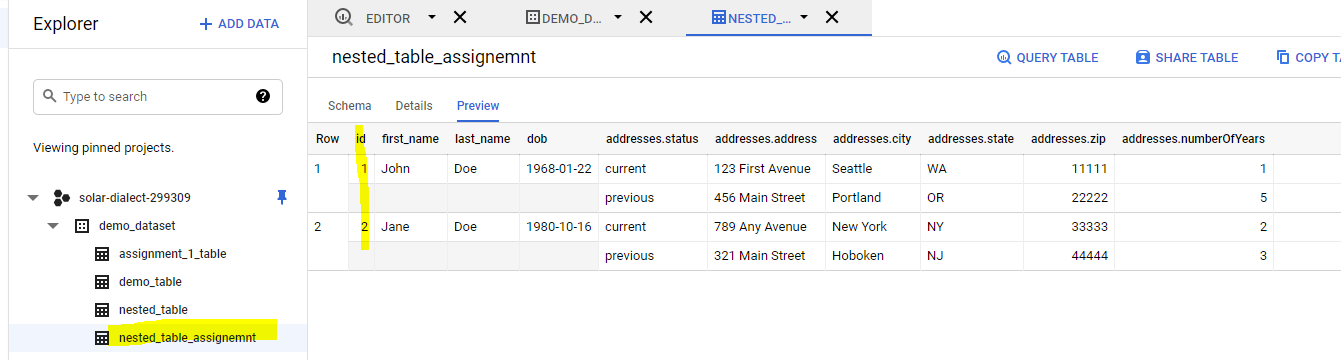


Table Is created



Write a query to fetch all columns where the city in address field is 'Portland'

SELECT id,first\_name,last\_name,dob, a FROM `solar-dialect-299309.demo\_dataset.nested\_table\_assignemnt`, unnest(addresses) as a where city="Portland"

