Extracting JSON Data

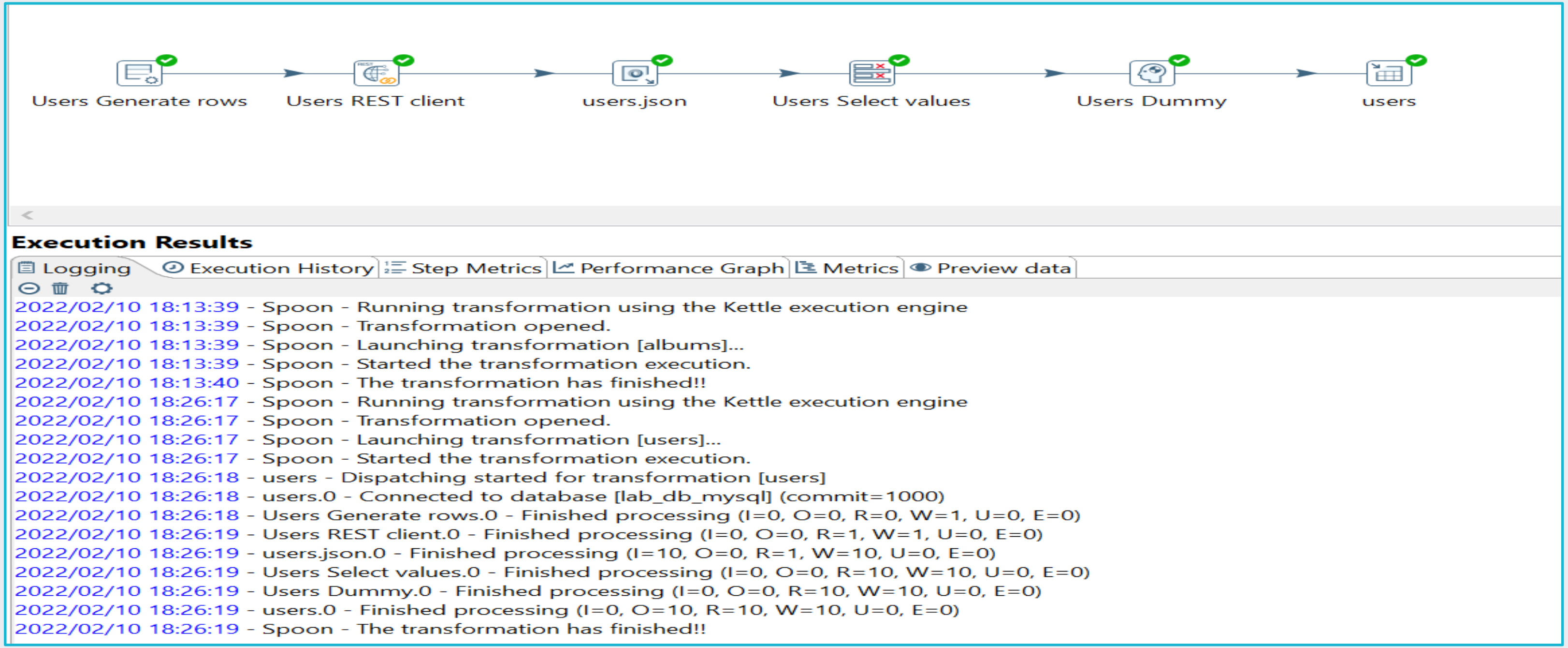
Kay Han

#1

Users

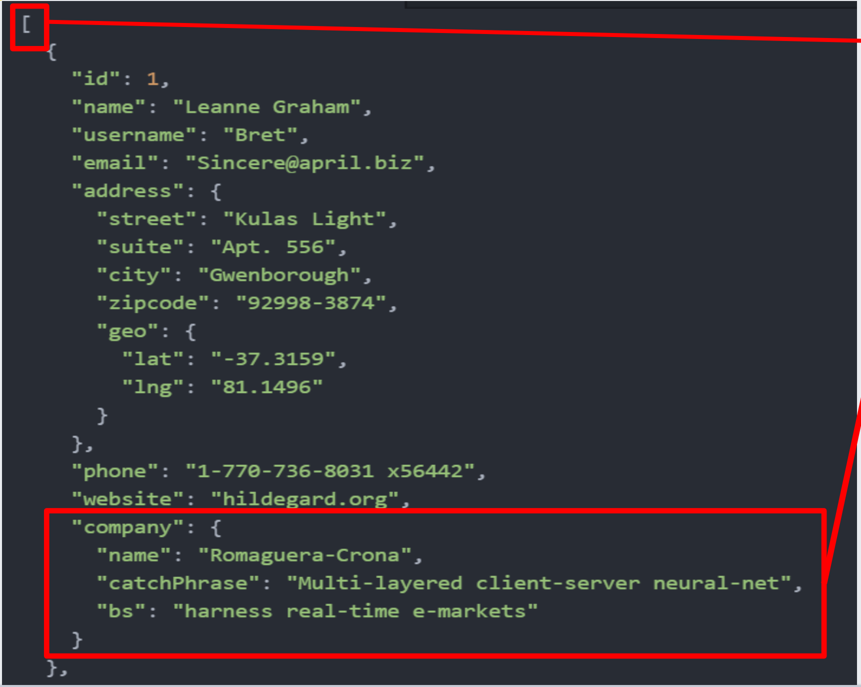
https://jsonplaceholder.typicode.com/users

Kettle Implementation



```
2022/02/10 18:13:39 - Spoon - Running transformation using the Kettle execution engine
2022/02/10 18:13:39 - Spoon - Transformation opened.
2022/02/10 18:13:39 - Spoon - Launching transformation [albums]...
2022/02/10 18:13:39 - Spoon - Started the transformation execution.
2022/02/10 18:13:40 - Spoon - The transformation has finished!!
2022/02/10 18:26:17 - Spoon - Running transformation using the Kettle execution engine
2022/02/10 18:26:17 - Spoon - Transformation opened.
2022/02/10 18:26:17 - Spoon - Launching transformation [users]...
2022/02/10 18:26:17 - Spoon - Started the transformation execution.
2022/02/10 18:26:18 - users - Dispatching started for transformation [users]
2022/02/10 18:26:18 - users.0 - Connected to database [lab_db_mysql] (commit=1000)
2022/02/10 18:26:18 - Users Generate rows.0 - Finished processing (I=0, O=0, R=0, W=1, U=0, E=0)
2022/02/10 18:26:19 - Users REST client.0 - Finished processing (I=0, O=0, R=1, W=1, U=0, E=0)
2022/02/10 18:26:19 - users.json.0 - Finished processing (I=10, O=0, R=1, W=10, U=0, E=0)
2022/02/10 18:26:19 - Users Select values.0 - Finished processing (I=0, O=0, R=10, W=10, U=0, E=0)
2022/02/10 18:26:19 - Users Dummy.0 - Finished processing (I=0, O=0, R=10, W=10, U=0, E=0)
2022/02/10 18:26:19 - users.0 - Finished processing (I=0, O=10, R=10, W=10, U=0, E=0)
2022/02/10 18:26:19 - Spoon - The transformation has finished!!
```

JSON Data: users.json



```
[
  {
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
    },
    "phone": "1-770-736-8031 x56442",
    "website": "hildegard.org",
    "company": {
      "name": "Romaguera-Crona",
      "catchPhrase": "Multi-layered client-server neural-net",
      "bs": "harness real-time e-markets"
    }
  },
```

1. The file is an array which is an ordered collection of values.
2. It has a nested structure with the same property name at the different levels.
3. Therefore, it should indicate a correct property in order to extract the data well.

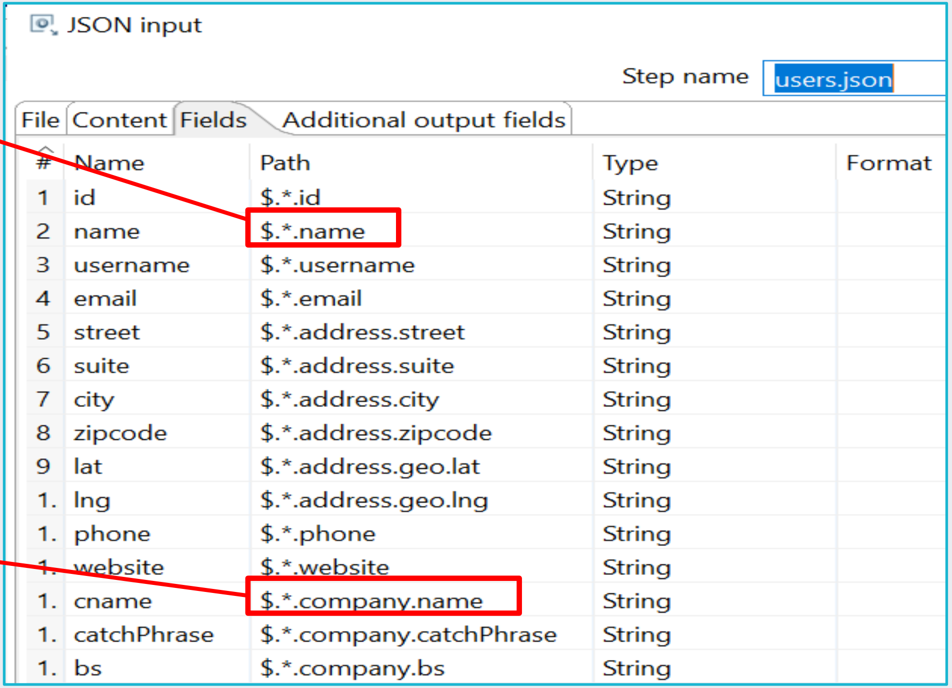
JSON Path

Operator	Description
$	The root element to query
*	Wildcard. All objects and/or elements
.<name>	Child operator. Dot-notated child.

JSON Input: Select Fields

Name property in an array from $root element.

Name property as a child property of Company property in a nested structure

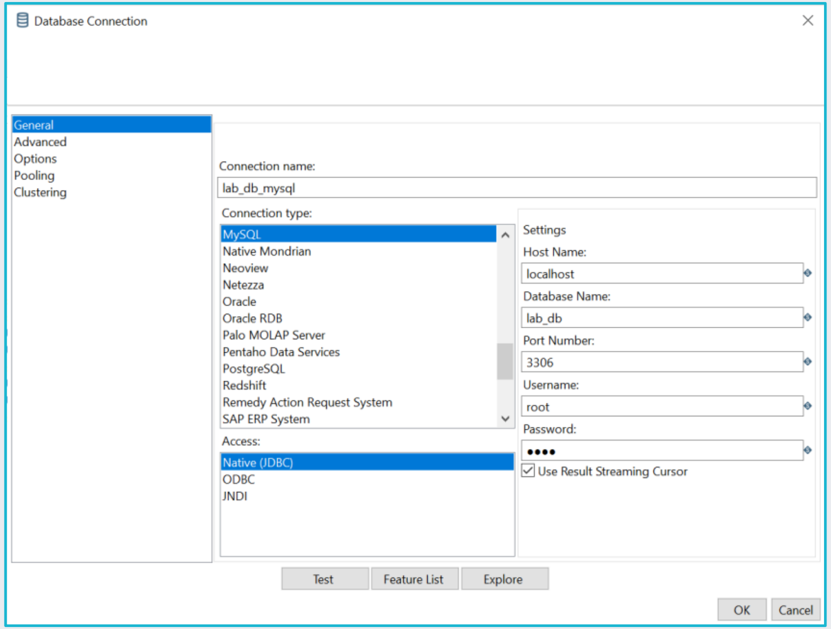


JSON input

Step name users.json

#	Name	Path	Type	Format
1	id	$.*.id	String	
2	name	$.*.name	String	
3	username	$.*.username	String	
4	email	$.*.email	String	
5	street	$.*.address.street	String	
6	suite	$.*.address.suite	String	
7	city	$.*.address.city	String	
8	zipcode	$.*.address.zipcode	String	
9	lat	$.*.address.geo.lat	String	
1.	lng	$.*.address.geo.lng	String	
1.	phone	$.*.phone	String	
1.	website	$.*.website	String	
1.	cname	$.*.company.name	String	
1.	catchPhrase	$.*.company.catchPhrase	String	
1.	bs	$.*.company.bs	String	

Database Connection & Table Output



Connection name: lab_db_mysql

Connection type: MySQL

Access: Native (JDBC)

Settings

Host Name: localhost

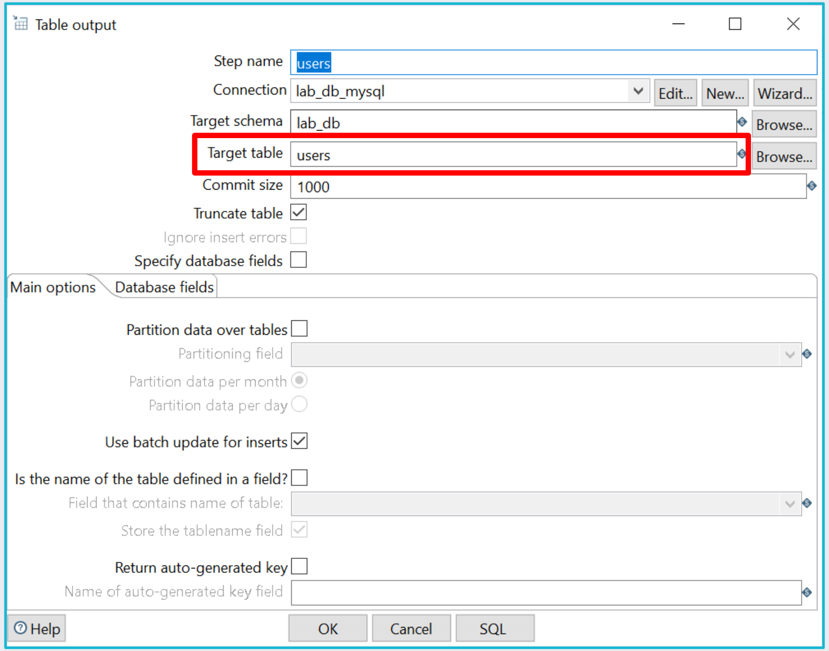
Database Name: lab_db

Port Number: 3306

Username: root

Password: ••••

☑ Use Result Streaming Cursor



Step name: users

Connection: lab_db_mysql

Target schema: lab_db

Target table: users

Commit size: 1000

Truncate table ☑

Ignore insert errors □

Specify database fields □

Partition data over tables □

Partitioning field

Partition data per month ●

Partition data per day ○

Use batch update for inserts ☑

Is the name of the table defined in a field? □

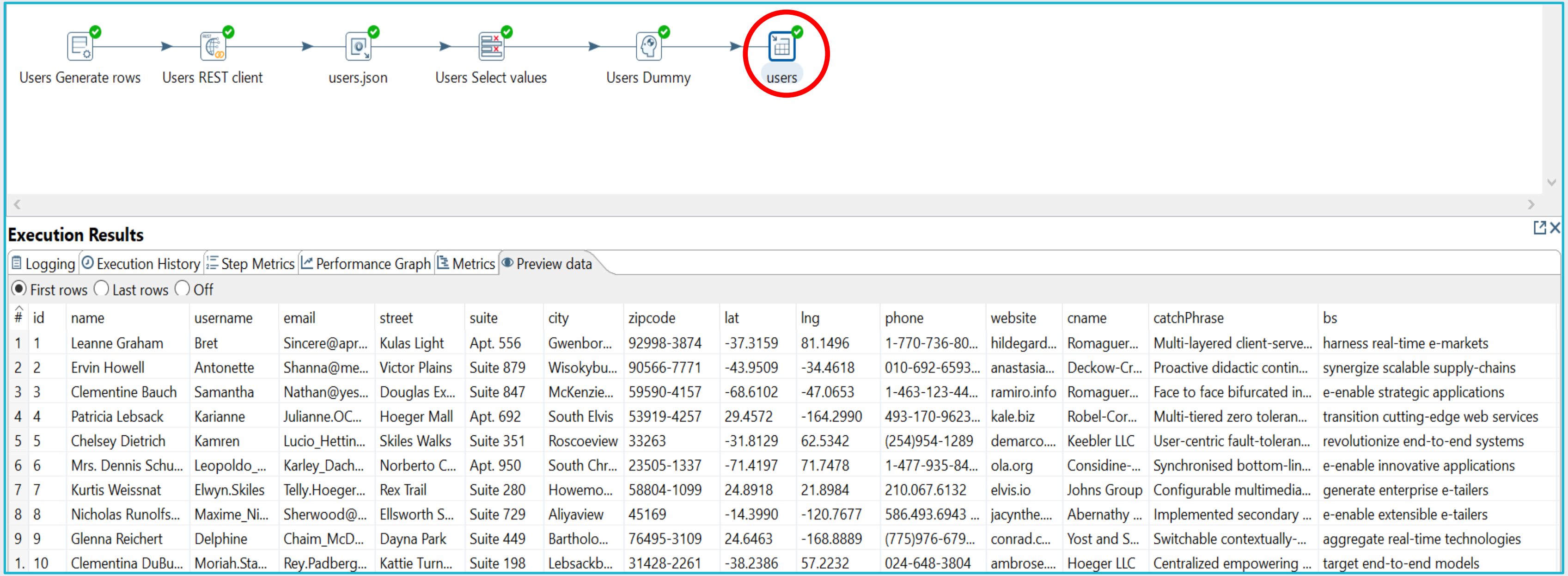
Field that contains name of table:

Store the tablename field ☑

Return auto-generated key □

Name of auto-generated key field

Preview Data (Users Table)

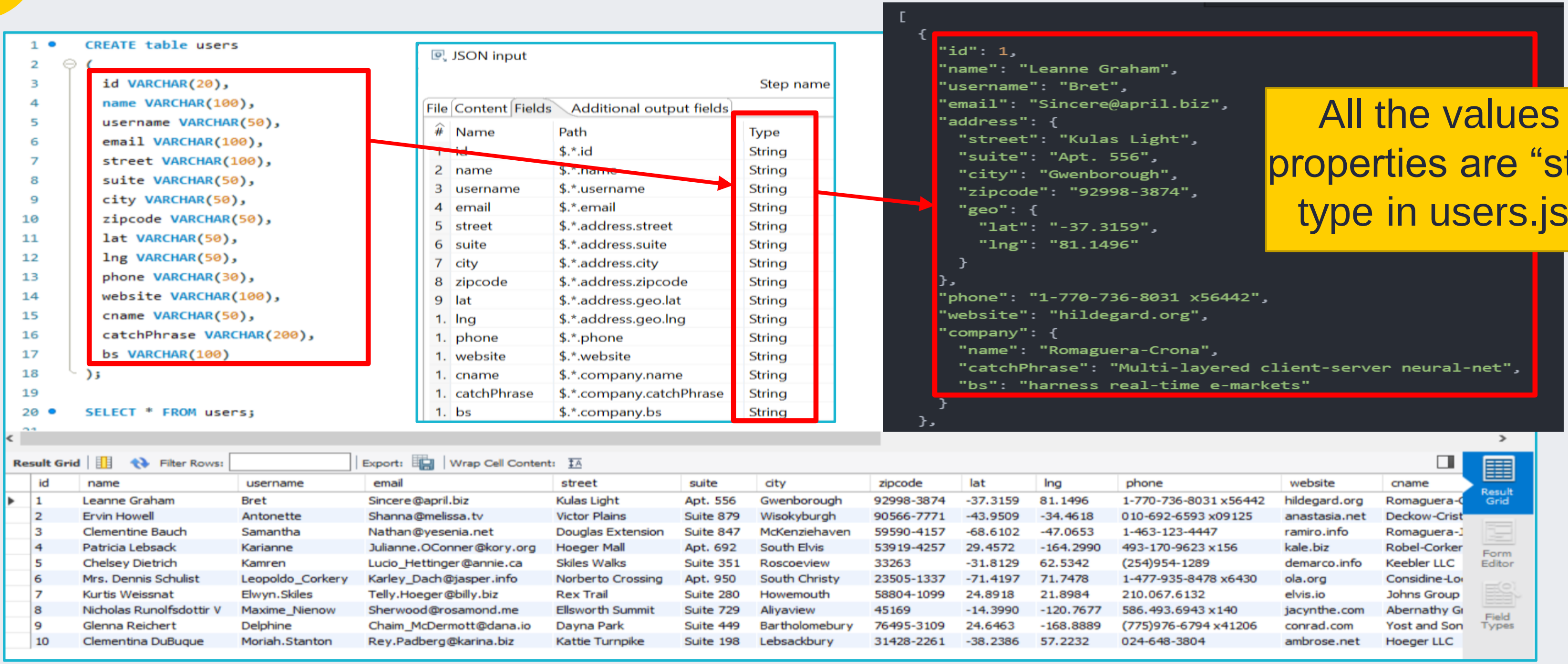


Execution Results

● First rows ○ Last rows ○ Off

#	id	name	username	email	street	suite	city	zipcode	lat	lng	phone	website	cname	catchPhrase	bs
1	1	Leanne Graham	Bret	Sincere@apr...	Kulas Light	Apt. 556	Gwenbor...	92998-3874	-37.3159	81.1496	1-770-736-80...	hildegard...	Romaguer...	Multi-layered client-serve...	harness real-time e-markets
2	2	Ervin Howell	Antonette	Shanna@me...	Victor Plains	Suite 879	Wisokybu...	90566-7771	-43.9509	-34.4618	010-692-6593...	anastasia...	Deckow-Cr...	Proactive didactic contin...	synergize scalable supply-chains
3	3	Clementine Bauch	Samantha	Nathan@yes...	Douglas Ex...	Suite 847	McKenzie...	59590-4157	-68.6102	-47.0653	1-463-123-44...	ramiro.info	Romaguer...	Face to face bifurcated in...	e-enable strategic applications
4	4	Patricia Lebsack	Karianne	Julianne.OC...	Hoeger Mall	Apt. 692	South Elvis	53919-4257	29.4572	-164.2990	493-170-9623...	kale.biz	Robel-Cor...	Multi-tiered zero toleran...	transition cutting-edge web services
5	5	Chelsey Dietrich	Kamren	Lucio_Hettin...	Skiles Walks	Suite 351	Roscoeview	33263	-31.8129	62.5342	(254)954-1289	demarco....	Keebler LLC	User-centric fault-toleran...	revolutionize end-to-end systems
6	6	Mrs. Dennis Schu...	Leopoldo_...	Karley_Dach...	Norberto C...	Apt. 950	South Chr...	23505-1337	-71.4197	71.7478	1-477-935-84...	ola.org	Considine-...	Synchronised bottom-lin...	e-enable innovative applications
7	7	Kurtis Weissnat	Elwyn.Skiles	Telly.Hoeger...	Rex Trail	Suite 280	Howemo...	58804-1099	24.8918	21.8984	210.067.6132	elvis.io	Johns Group	Configurable multimedia...	generate enterprise e-tailers
8	8	Nicholas Runolfs...	Maxime_Ni...	Sherwood@...	Ellsworth S...	Suite 729	Aliyaview	45169	-14.3990	-120.7677	586.493.6943 ...	jacynthe....	Abernathy ...	Implemented secondary ...	e-enable extensible e-tailers
9	9	Glenna Reichert	Delphine	Chaim_McD...	Dayna Park	Suite 449	Bartholo...	76495-3109	24.6463	-168.8889	(775)976-679...	conrad.c...	Yost and S...	Switchable contextually-...	aggregate real-time technologies
1.	10	Clementina DuBu...	Moriah.Sta...	Rey.Padberg...	Kattie Turn...	Suite 198	Lebsackb...	31428-2261	-38.2386	57.2232	024-648-3804	ambrose....	Hoeger LLC	Centralized empowering ...	target end-to-end models

MySQL Workbench: Users Table



```sql
CREATE table users
(
    id VARCHAR(20),
    name VARCHAR(100),
    username VARCHAR(50),
    email VARCHAR(100),
    street VARCHAR(100),
    suite VARCHAR(50),
    city VARCHAR(50),
    zipcode VARCHAR(50),
    lat VARCHAR(50),
    lng VARCHAR(50),
    phone VARCHAR(30),
    website VARCHAR(100),
    cname VARCHAR(50),
    catchPhrase VARCHAR(200),
    bs VARCHAR(100)
);

SELECT * FROM users;
```

JSON input

Step name

File | Content | Fields | Additional output fields

#	Name	Path	Type
1	id	$.*.id	String
2	name	$.*.name	String
3	username	$.*.username	String
4	email	$.*.email	String
5	street	$.*.address.street	String
6	suite	$.*.address.suite	String
7	city	$.*.address.city	String
8	zipcode	$.*.address.zipcode	String
9	lat	$.*.address.geo.lat	String
1.	lng	$.*.address.geo.lng	String
1.	phone	$.*.phone	String
1.	website	$.*.website	String
1.	cname	$.*.company.name	String
1.	catchPhrase	$.*.company.catchPhrase	String
1.	bs	$.*.company.bs	String

```json
[
  {
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
    },
    "phone": "1-770-736-8031 x56442",
    "website": "hildegard.org",
    "company": {
      "name": "Romaguera-Crona",
      "catchPhrase": "Multi-layered client-server neural-net",
      "bs": "harness real-time e-markets"
    }
  },
```

All the values of properties are "string" type in users.json.

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

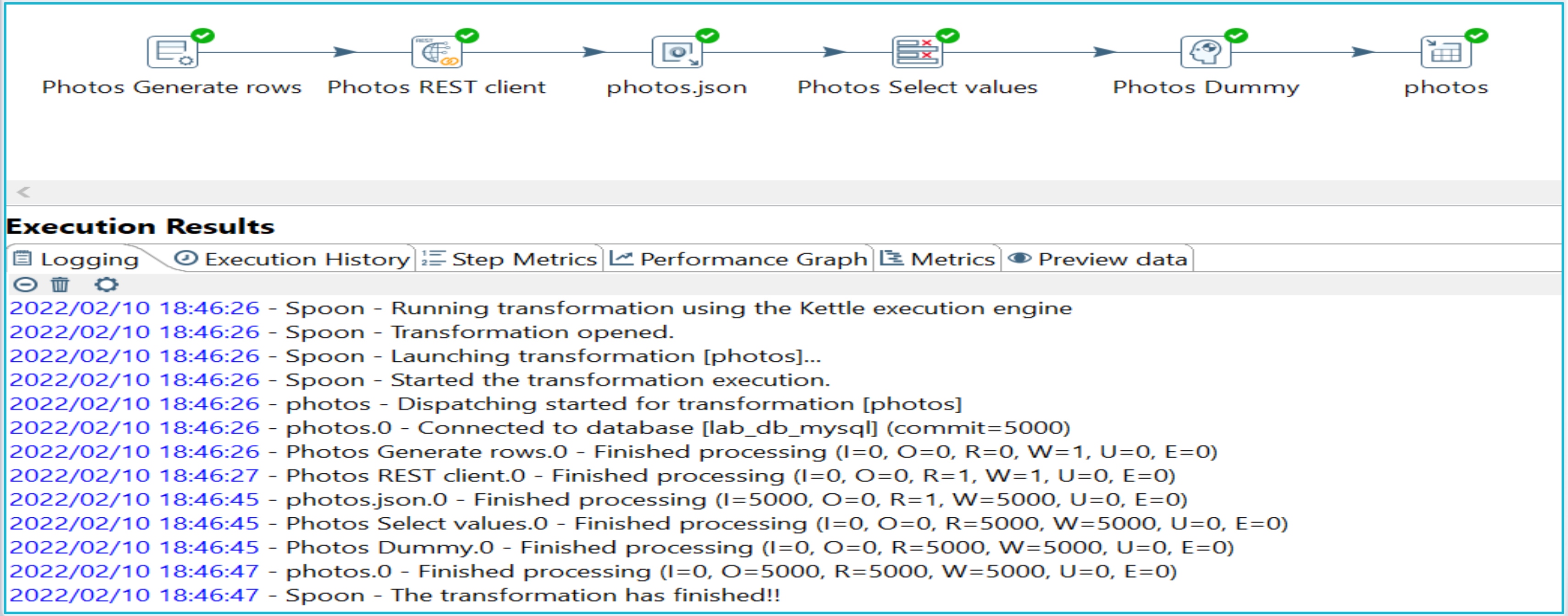
id	name	username	email	street	suite	city	zipcode	lat	lng	phone	website	cname
1	Leanne Graham	Bret	Sincere@april.biz	Kulas Light	Apt. 556	Gwenborough	92998-3874	-37.3159	81.1496	1-770-736-8031 x56442	hildegard.org	Romaguera-(
2	Ervin Howell	Antonette	Shanna@melissa.tv	Victor Plains	Suite 879	Wisokyburgh	90566-7771	-43.9509	-34.4618	010-692-6593 x09125	anastasia.net	Deckow-Crist
3	Clementine Bauch	Samantha	Nathan@yesenia.net	Douglas Extension	Suite 847	McKenziehaven	59590-4157	-68.6102	-47.0653	1-463-123-4447	ramiro.info	Romaguera-J
4	Patricia Lebsack	Karianne	Julianne.OConner@kory.org	Hoeger Mall	Apt. 692	South Elvis	53919-4257	29.4572	-164.2990	493-170-9623 x156	kale.biz	Robel-Corker
5	Chelsey Dietrich	Kamren	Lucio_Hettinger@annie.ca	Skiles Walks	Suite 351	Roscoeview	33263	-31.8129	62.5342	(254)954-1289	demarco.info	Keebler LLC
6	Mrs. Dennis Schulist	Leopoldo_Corkery	Karley_Dach@jasper.info	Norberto Crossing	Apt. 950	South Christy	23505-1337	-71.4197	71.7478	1-477-935-8478 x6430	ola.org	Considine-Lo
7	Kurtis Weissnat	Elwyn.Skiles	Telly.Hoeger@billy.biz	Rex Trail	Suite 280	Howemouth	58804-1099	24.8918	21.8984	210.067.6132	elvis.io	Johns Group
8	Nicholas Runolfsdottir V	Maxime_Nienow	Sherwood@rosamond.me	Ellsworth Summit	Suite 729	Aliyaview	45169	-14.3990	-120.7677	586.493.6943 x140	jacynthe.com	Abernathy G
9	Glenna Reichert	Delphine	Chaim_McDermott@dana.io	Dayna Park	Suite 449	Bartholomebury	76495-3109	24.6463	-168.8889	(775)976-6794 x41206	conrad.com	Yost and Son
10	Clementina DuBuque	Moriah.Stanton	Rey.Padberg@karina.biz	Kattie Turnpike	Suite 198	Lebsackbury	31428-2261	-38.2386	57.2232	024-648-3804	ambrose.net	Hoeger LLC

#2

Photos

https://jsonplaceholder.typicode.com/photos

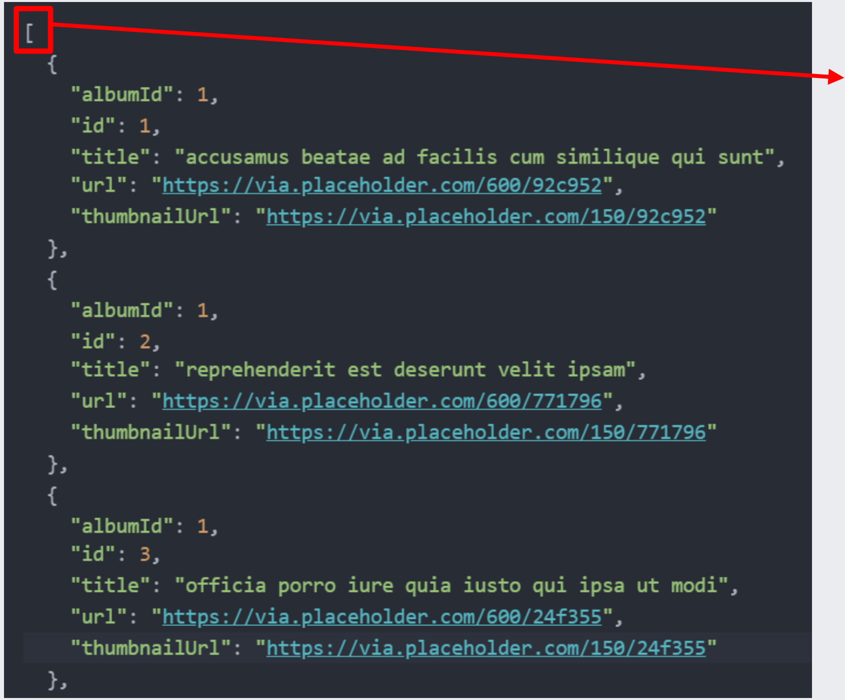
Kettle Implementation



Execution Results

```
2022/02/10 18:46:26 - Spoon - Running transformation using the Kettle execution engine
2022/02/10 18:46:26 - Spoon - Transformation opened.
2022/02/10 18:46:26 - Spoon - Launching transformation [photos]...
2022/02/10 18:46:26 - Spoon - Started the transformation execution.
2022/02/10 18:46:26 - photos - Dispatching started for transformation [photos]
2022/02/10 18:46:26 - photos.0 - Connected to database [lab_db_mysql] (commit=5000)
2022/02/10 18:46:26 - Photos Generate rows.0 - Finished processing (I=0, O=0, R=0, W=1, U=0, E=0)
2022/02/10 18:46:27 - Photos REST client.0 - Finished processing (I=0, O=0, R=1, W=1, U=0, E=0)
2022/02/10 18:46:45 - photos.json.0 - Finished processing (I=5000, O=0, R=1, W=5000, U=0, E=0)
2022/02/10 18:46:45 - Photos Select values.0 - Finished processing (I=0, O=0, R=5000, W=5000, U=0, E=0)
2022/02/10 18:46:45 - Photos Dummy.0 - Finished processing (I=0, O=0, R=5000, W=5000, U=0, E=0)
2022/02/10 18:46:47 - photos.0 - Finished processing (I=0, O=5000, R=5000, W=5000, U=0, E=0)
2022/02/10 18:46:47 - Spoon - The transformation has finished!!
```

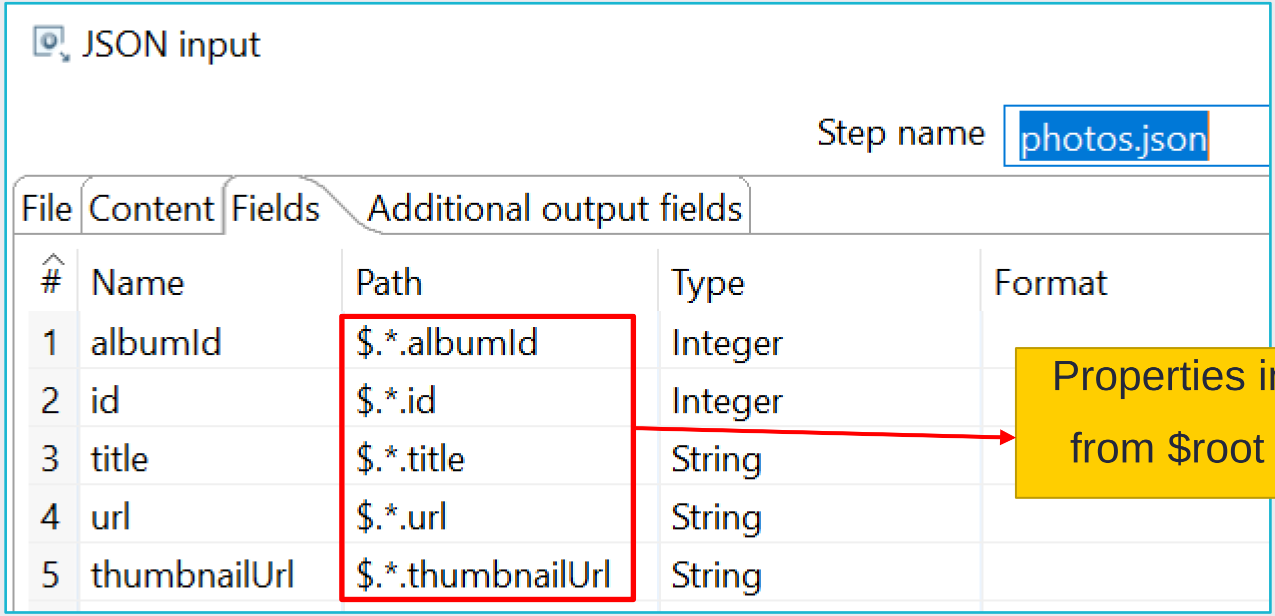
JSON Data: photos.json



```
[
  {
    "albumId": 1,
    "id": 1,
    "title": "accusamus beatae ad facilis cum similique qui sunt",
    "url": "https://via.placeholder.com/600/92c952",
    "thumbnailUrl": "https://via.placeholder.com/150/92c952"
  },
  {
    "albumId": 1,
    "id": 2,
    "title": "reprehenderit est deserunt velit ipsam",
    "url": "https://via.placeholder.com/600/771796",
    "thumbnailUrl": "https://via.placeholder.com/150/771796"
  },
  {
    "albumId": 1,
    "id": 3,
    "title": "officia porro iure quia iusto qui ipsa ut modi",
    "url": "https://via.placeholder.com/600/24f355",
    "thumbnailUrl": "https://via.placeholder.com/150/24f355"
  },
```

The file is an array which is an ordered collection of values.

JSON Input: Select Fields



Step name: photos.json

#	Name	Path	Type	Format
1	albumId	$.*.albumId	Integer	
2	id	$.*.id	Integer	
3	title	$.*.title	String	
4	url	$.*.url	String	
5	thumbnailUrl	$.*.thumbnailUrl	String	

Properties in an array from $root element.

Table Output

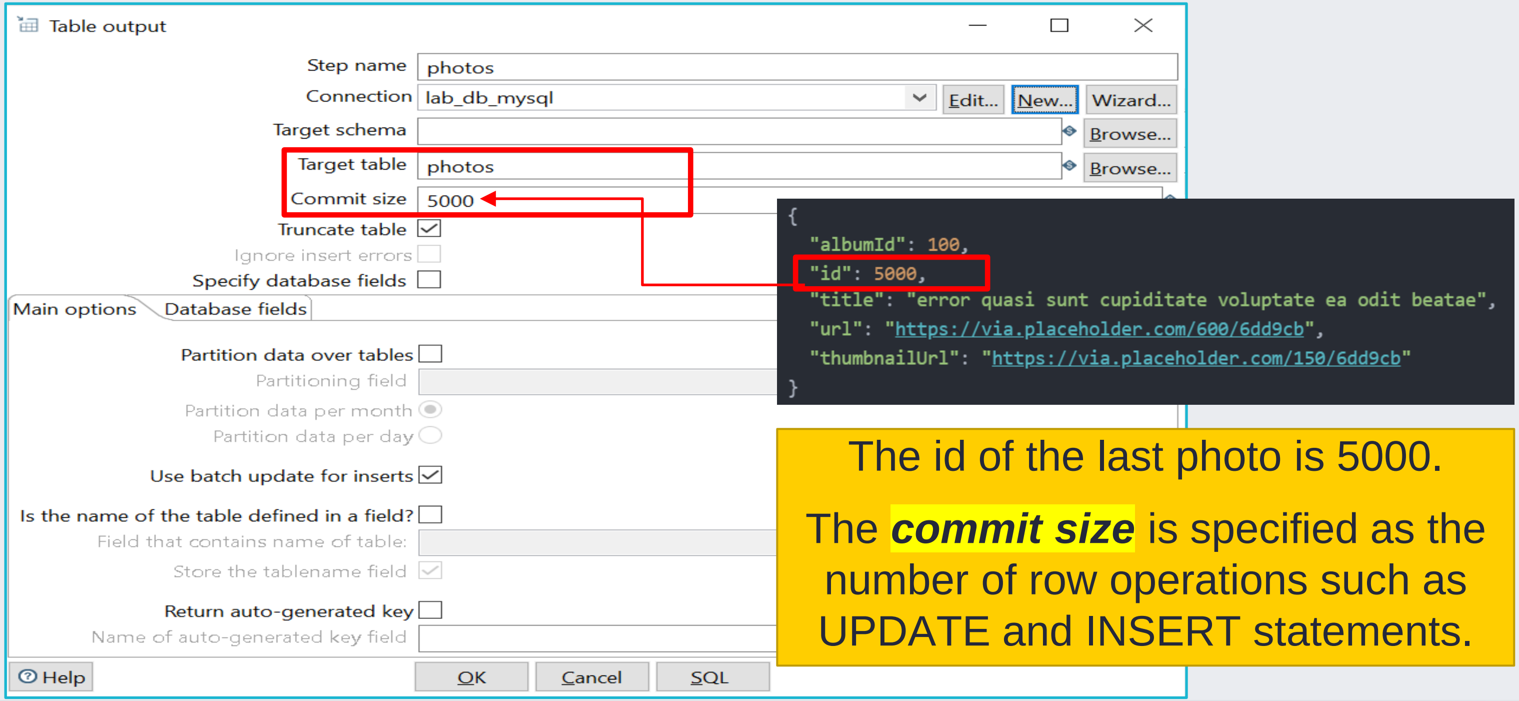


Table output

Step name: photos

Connection: lab_db_mysql

Target schema:

Target table: photos

Commit size: 5000

Truncate table ☑

Ignore insert errors □

Specify database fields □

Partition data over tables □

Partitioning field

Partition data per month ●

Partition data per day ○

Use batch update for inserts ☑

Is the name of the table defined in a field? □

Field that contains name of table:

Store the tablename field ☑

Return auto-generated key □

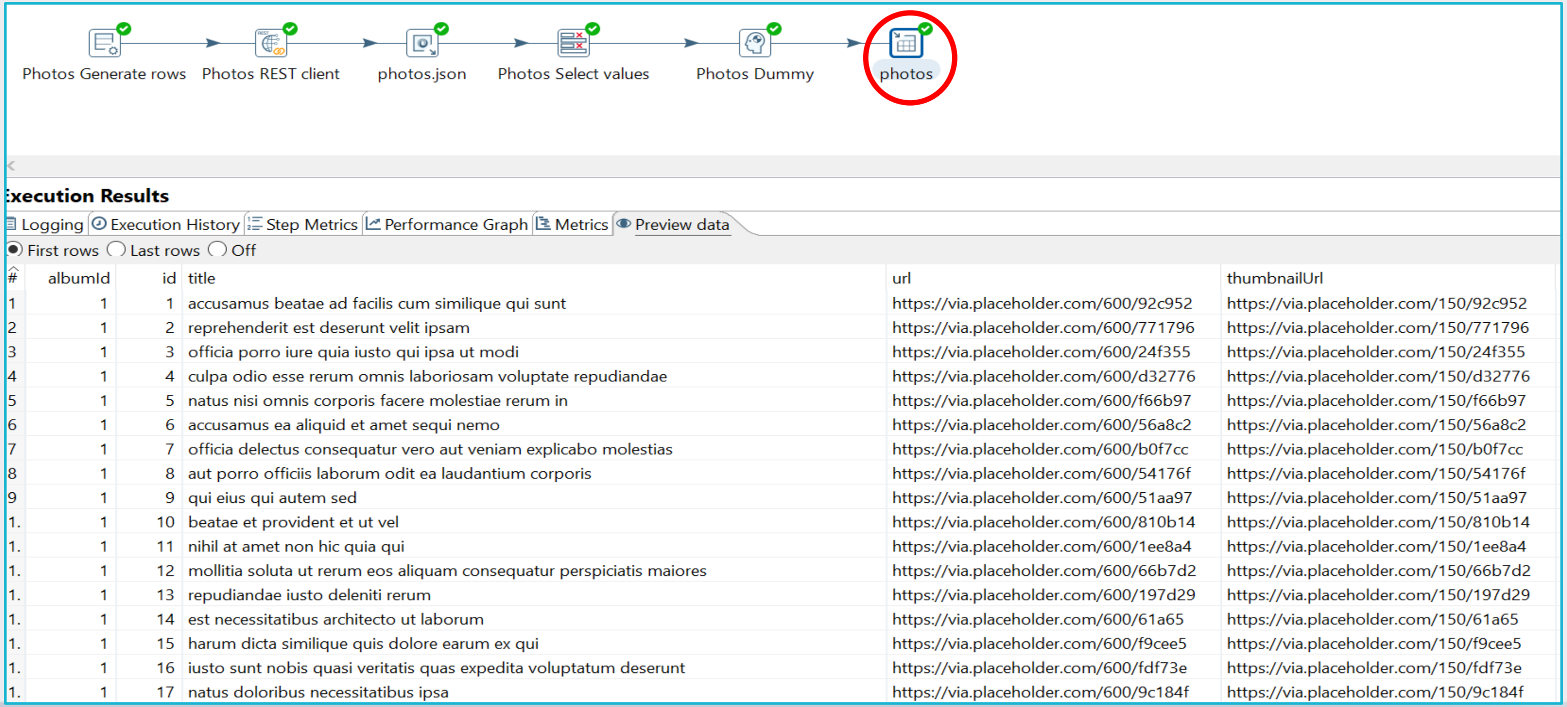
Name of auto-generated key field

```
{
  "albumId": 100,
  "id": 5000,
  "title": "error quasi sunt cupiditate voluptate ea odit beatae",
  "url": "https://via.placeholder.com/600/6dd9cb",
  "thumbnailUrl": "https://via.placeholder.com/150/6dd9cb"
}
```

The id of the last photo is 5000.

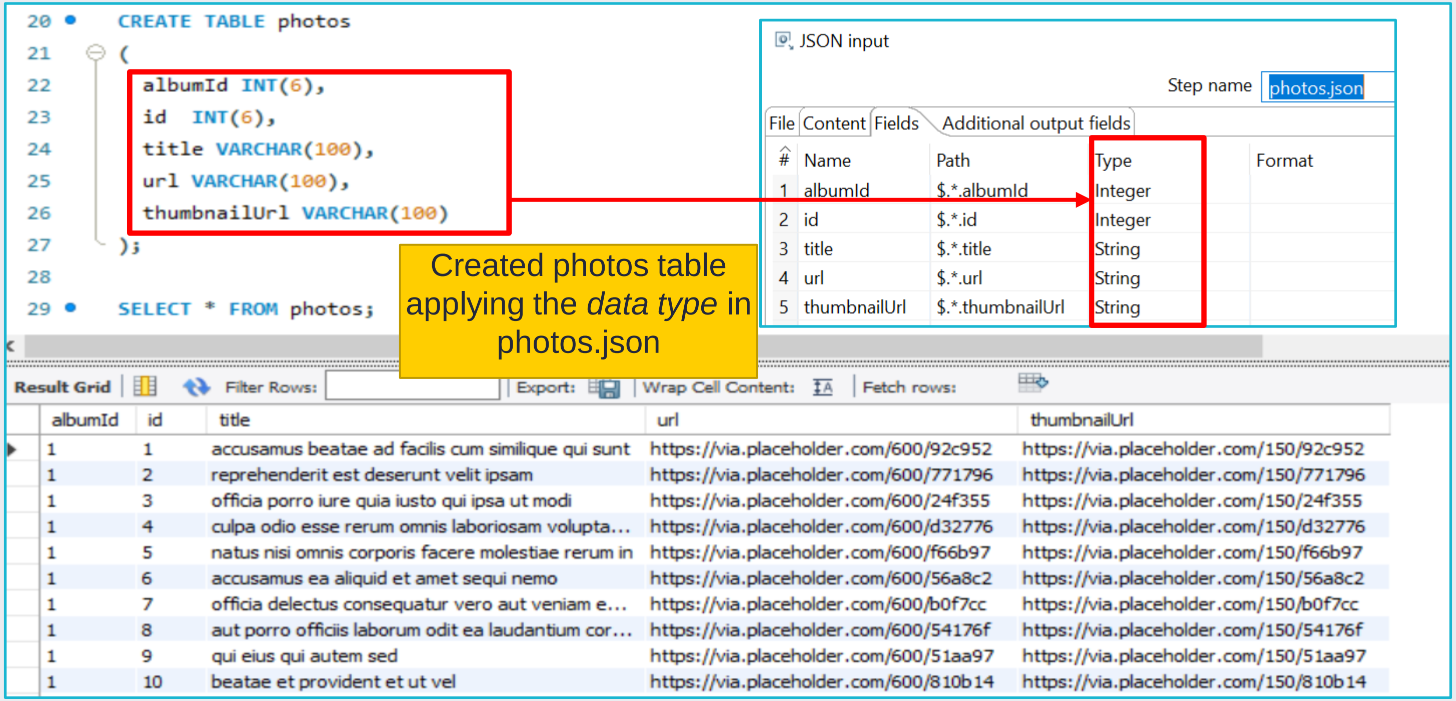
The ***commit size*** is specified as the number of row operations such as UPDATE and INSERT statements.

Preview Data (Photos Table)



#	albumId	id	title	url	thumbnailUrl
1	1	1	accusamus beatae ad facilis cum similique qui sunt	https://via.placeholder.com/600/92c952	https://via.placeholder.com/150/92c952
2	1	2	reprehenderit est deserunt velit ipsam	https://via.placeholder.com/600/771796	https://via.placeholder.com/150/771796
3	1	3	officia porro iure quia iusto qui ipsa ut modi	https://via.placeholder.com/600/24f355	https://via.placeholder.com/150/24f355
4	1	4	culpa odio esse rerum omnis laboriosam voluptate repudiandae	https://via.placeholder.com/600/d32776	https://via.placeholder.com/150/d32776
5	1	5	natus nisi omnis corporis facere molestiae rerum in	https://via.placeholder.com/600/f66b97	https://via.placeholder.com/150/f66b97
6	1	6	accusamus ea aliquid et amet sequi nemo	https://via.placeholder.com/600/56a8c2	https://via.placeholder.com/150/56a8c2
7	1	7	officia delectus consequatur vero aut veniam explicabo molestias	https://via.placeholder.com/600/b0f7cc	https://via.placeholder.com/150/b0f7cc
8	1	8	aut porro officiis laborum odit ea laudantium corporis	https://via.placeholder.com/600/54176f	https://via.placeholder.com/150/54176f
9	1	9	qui eius qui autem sed	https://via.placeholder.com/600/51aa97	https://via.placeholder.com/150/51aa97
1.	1	10	beatae et provident et ut vel	https://via.placeholder.com/600/810b14	https://via.placeholder.com/150/810b14
1.	1	11	nihil at amet non hic quia qui	https://via.placeholder.com/600/1ee8a4	https://via.placeholder.com/150/1ee8a4
1.	1	12	mollitia soluta ut rerum eos aliquam consequatur perspiciatis maiores	https://via.placeholder.com/600/66b7d2	https://via.placeholder.com/150/66b7d2
1.	1	13	repudiandae iusto deleniti rerum	https://via.placeholder.com/600/197d29	https://via.placeholder.com/150/197d29
1.	1	14	est necessitatibus architecto ut laborum	https://via.placeholder.com/600/61a65	https://via.placeholder.com/150/61a65
1.	1	15	harum dicta similique quis dolore earum ex qui	https://via.placeholder.com/600/f9cee5	https://via.placeholder.com/150/f9cee5
1.	1	16	iusto sunt nobis quasi veritatis quas expedita voluptatum deserunt	https://via.placeholder.com/600/fdf73e	https://via.placeholder.com/150/fdf73e
1.	1	17	natus doloribus necessitatibus ipsa	https://via.placeholder.com/600/9c184f	https://via.placeholder.com/150/9c184f

MySQL Workbench: Photos Table



```
CREATE TABLE photos
(
    albumId INT(6),
    id  INT(6),
    title VARCHAR(100),
    url VARCHAR(100),
    thumbnailUrl VARCHAR(100)
);

SELECT * FROM photos;
```

JSON input

Step name: photos.json

#	Name	Path	Type	Format
1	albumId	$.*.albumId	Integer	
2	id	$.*.id	Integer	
3	title	$.*.title	String	
4	url	$.*.url	String	
5	thumbnailUrl	$.*.thumbnailUrl	String	

Created photos table applying the *data type* in photos.json

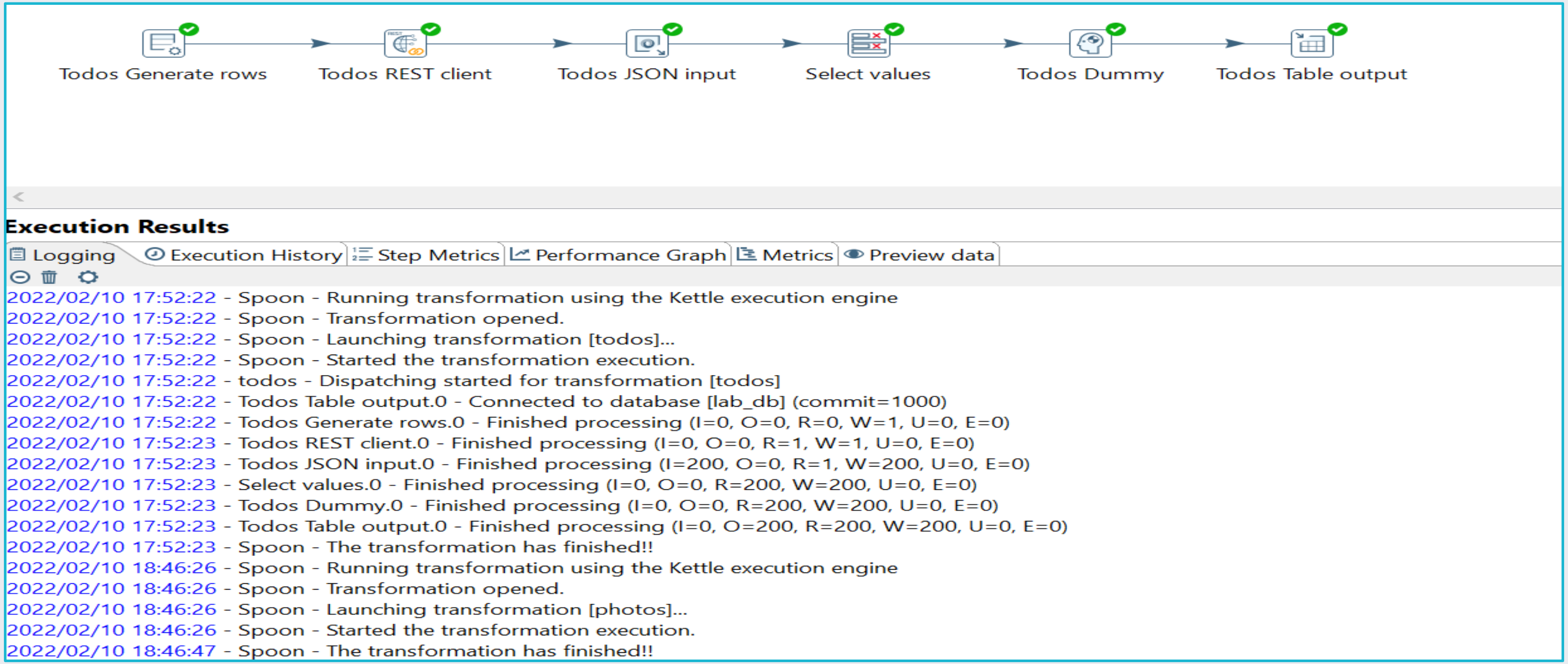
albumId	id	title	url	thumbnailUrl
1	1	accusamus beatae ad facilis cum similique qui sunt	https://via.placeholder.com/600/92c952	https://via.placeholder.com/150/92c952
1	2	reprehenderit est deserunt velit ipsam	https://via.placeholder.com/600/771796	https://via.placeholder.com/150/771796
1	3	officia porro iure quia iusto qui ipsa ut modi	https://via.placeholder.com/600/24f355	https://via.placeholder.com/150/24f355
1	4	culpa odio esse rerum omnis laboriosam volupta...	https://via.placeholder.com/600/d32776	https://via.placeholder.com/150/d32776
1	5	natus nisi omnis corporis facere molestiae rerum in	https://via.placeholder.com/600/f66b97	https://via.placeholder.com/150/f66b97
1	6	accusamus ea aliquid et amet sequi nemo	https://via.placeholder.com/600/56a8c2	https://via.placeholder.com/150/56a8c2
1	7	officia delectus consequatur vero aut veniam e...	https://via.placeholder.com/600/b0f7cc	https://via.placeholder.com/150/b0f7cc
1	8	aut porro officiis laborum odit ea laudantium cor...	https://via.placeholder.com/600/54176f	https://via.placeholder.com/150/54176f
1	9	qui eius qui autem sed	https://via.placeholder.com/600/51aa97	https://via.placeholder.com/150/51aa97
1	10	beatae et provident et ut vel	https://via.placeholder.com/600/810b14	https://via.placeholder.com/150/810b14

#3

Todos

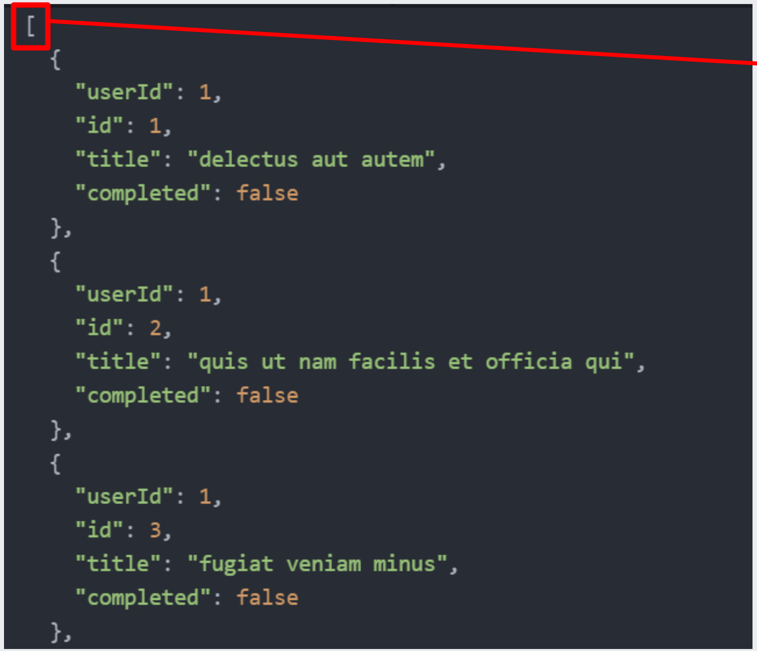
https://jsonplaceholder.typicode.com/todos

Kettle Implementation



```
2022/02/10 17:52:22 - Spoon - Running transformation using the Kettle execution engine
2022/02/10 17:52:22 - Spoon - Transformation opened.
2022/02/10 17:52:22 - Spoon - Launching transformation [todos]...
2022/02/10 17:52:22 - Spoon - Started the transformation execution.
2022/02/10 17:52:22 - todos - Dispatching started for transformation [todos]
2022/02/10 17:52:22 - Todos Table output.0 - Connected to database [lab_db] (commit=1000)
2022/02/10 17:52:22 - Todos Generate rows.0 - Finished processing (I=0, O=0, R=0, W=1, U=0, E=0)
2022/02/10 17:52:23 - Todos REST client.0 - Finished processing (I=0, O=0, R=1, W=1, U=0, E=0)
2022/02/10 17:52:23 - Todos JSON input.0 - Finished processing (I=200, O=0, R=1, W=200, U=0, E=0)
2022/02/10 17:52:23 - Select values.0 - Finished processing (I=0, O=0, R=200, W=200, U=0, E=0)
2022/02/10 17:52:23 - Todos Dummy.0 - Finished processing (I=0, O=0, R=200, W=200, U=0, E=0)
2022/02/10 17:52:23 - Todos Table output.0 - Finished processing (I=0, O=200, R=200, W=200, U=0, E=0)
2022/02/10 17:52:23 - Spoon - The transformation has finished!!
2022/02/10 18:46:26 - Spoon - Running transformation using the Kettle execution engine
2022/02/10 18:46:26 - Spoon - Transformation opened.
2022/02/10 18:46:26 - Spoon - Launching transformation [photos]...
2022/02/10 18:46:26 - Spoon - Started the transformation execution.
2022/02/10 18:46:47 - Spoon - The transformation has finished!!
```

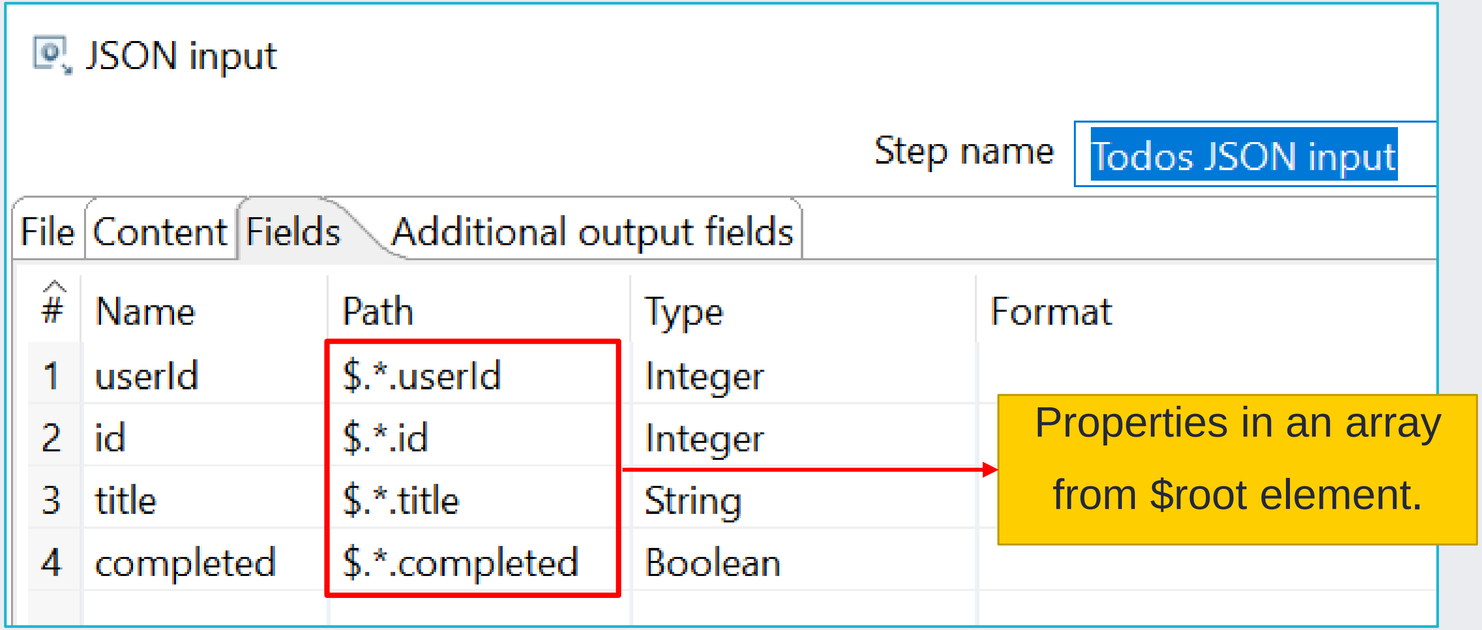
JSON Data: todos.json



```
[
  {
    "userId": 1,
    "id": 1,
    "title": "delectus aut autem",
    "completed": false
  },
  {
    "userId": 1,
    "id": 2,
    "title": "quis ut nam facilis et officia qui",
    "completed": false
  },
  {
    "userId": 1,
    "id": 3,
    "title": "fugiat veniam minus",
    "completed": false
  },
```

The file is an array which is an ordered collection of values.

JSON Input



JSON input

Step name: Todos JSON input

#	Name	Path	Type	Format
1	userId	$.*.userId	Integer	
2	id	$.*.id	Integer	
3	title	$.*.title	String	
4	completed	$.*.completed	Boolean	

Properties in an array from $root element.

Table Output

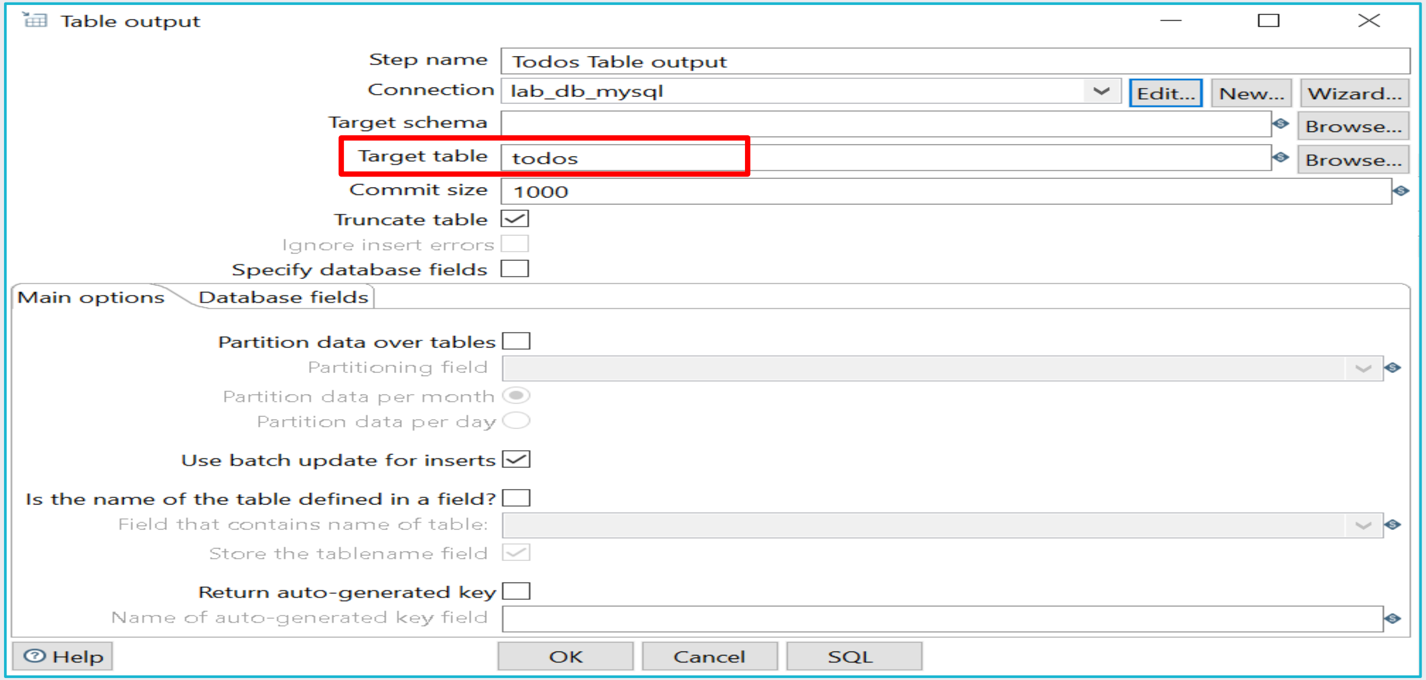


Table output

Step name: Todos Table output

Connection: lab_db_mysql

Target schema:

Target table: todos

Commit size: 1000

Truncate table ☑

Ignore insert errors □

Specify database fields □

Main options

Partition data over tables □

Partitioning field:

Partition data per month ●

Partition data per day ○

Use batch update for inserts ☑

Is the name of the table defined in a field? □

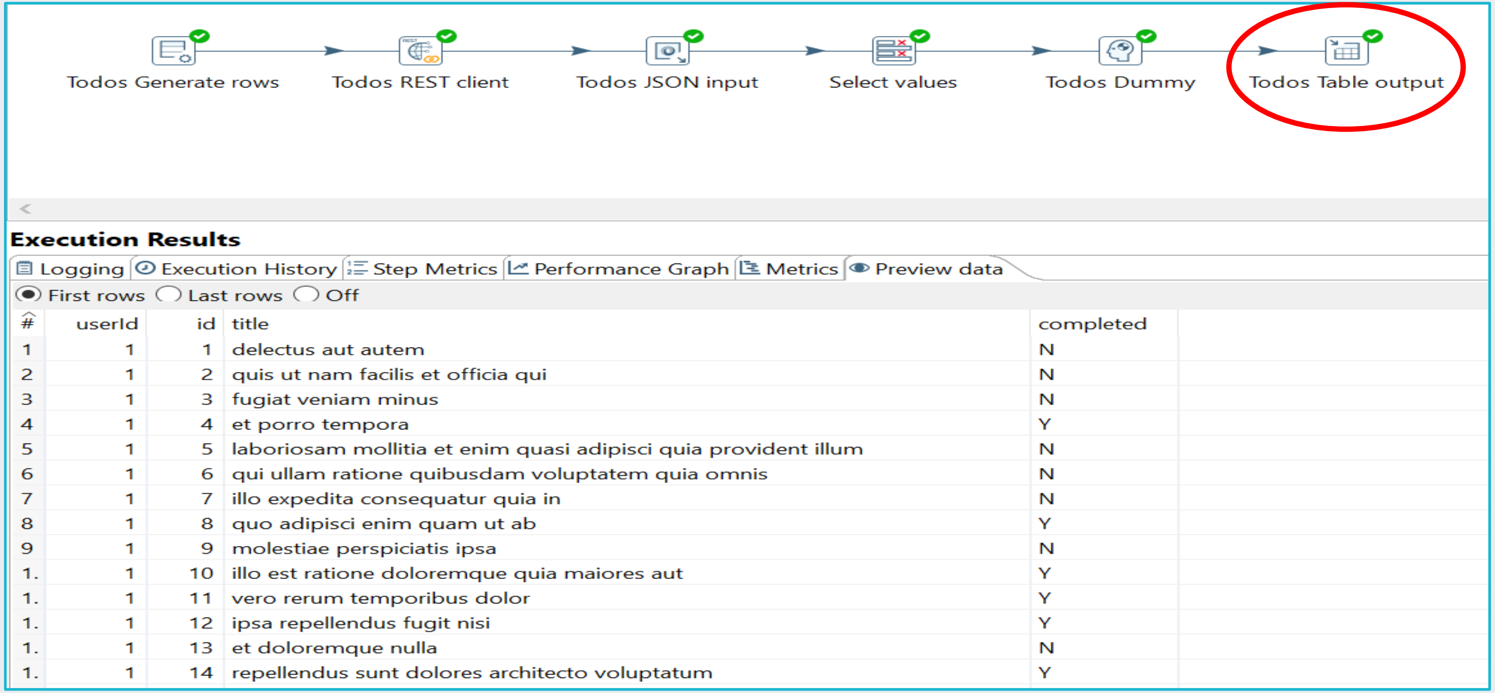
Field that contains name of table:

Store the tablename field ☑

Return auto-generated key □

Name of auto-generated key field:

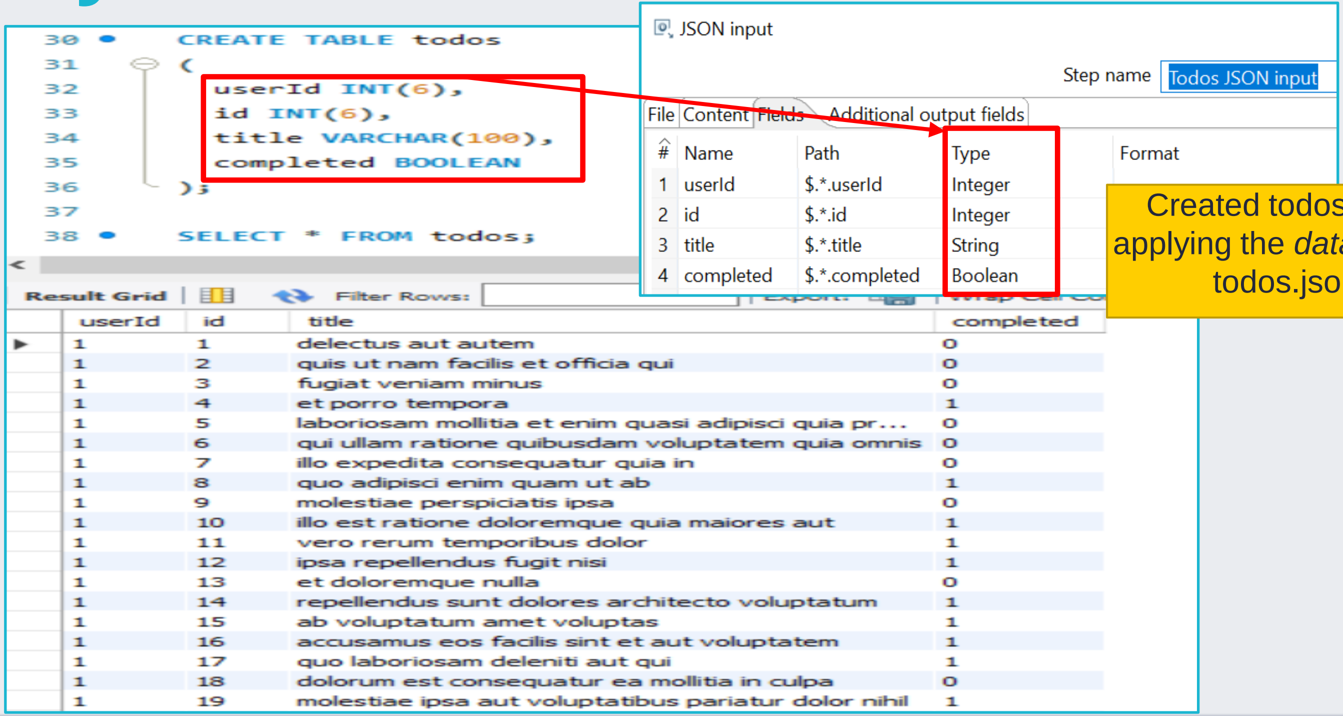
Preview Data (Todos Table)



Execution Results

#	userId	id	title	completed
1	1	1	delectus aut autem	N
2	1	2	quis ut nam facilis et officia qui	N
3	1	3	fugiat veniam minus	N
4	1	4	et porro tempora	Y
5	1	5	laboriosam mollitia et enim quasi adipisci quia provident illum	N
6	1	6	qui ullam ratione quibusdam voluptatem quia omnis	N
7	1	7	illo expedita consequatur quia in	N
8	1	8	quo adipisci enim quam ut ab	Y
9	1	9	molestiae perspiciatis ipsa	N
1.	1	10	illo est ratione doloremque quia maiores aut	Y
1.	1	11	vero rerum temporibus dolor	Y
1.	1	12	ipsa repellendus fugit nisi	Y
1.	1	13	et doloremque nulla	N
1.	1	14	repellendus sunt dolores architecto voluptatum	Y

MySQL Workbench: Todos



```
CREATE TABLE todos
(
    userId INT(6),
    id INT(6),
    title VARCHAR(100),
    completed BOOLEAN
);

SELECT * FROM todos;
```

JSON input

Step name: Todos JSON input

#	Name	Path	Type	Format
1	userId	$.*.userId	Integer	
2	id	$.*.id	Integer	
3	title	$.*.title	String	
4	completed	$.*.completed	Boolean	

userId	id	title	completed
1	1	delectus aut autem	0
1	2	quis ut nam facilis et officia qui	0
1	3	fugiat veniam minus	0
1	4	et porro tempora	1
1	5	laboriosam mollitia et enim quasi adipisci quia pr...	0
1	6	qui ullam ratione quibusdam voluptatem quia omnis	0
1	7	illo expedita consequatur quia in	0
1	8	quo adipisci enim quam ut ab	1
1	9	molestiae perspiciatis ipsa	0
1	10	illo est ratione doloremque quia maiores aut	1
1	11	vero rerum temporibus dolor	1
1	12	ipsa repellendus fugit nisi	1
1	13	et doloremque nulla	0
1	14	repellendus sunt dolores architecto voluptatum	1
1	15	ab voluptatum amet voluptas	1
1	16	accusamus eos facilis sint et aut voluptatem	1
1	17	quo laboriosam deleniti aut qui	1
1	18	dolorum est consequatur ea mollitia in culpa	0
1	19	molestiae ipsa aut voluptatibus pariatur dolor nihil	1

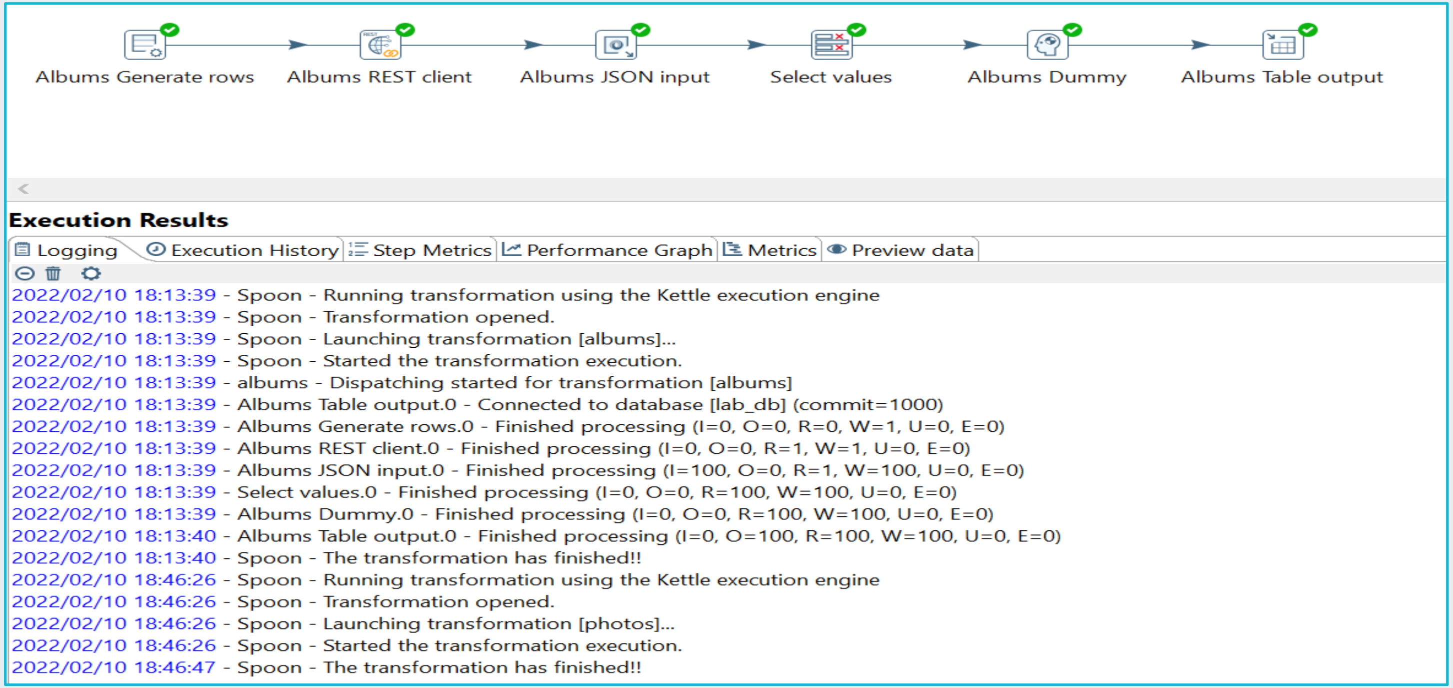
Created todos table applying the *data type* in todos.json

#4

Albums

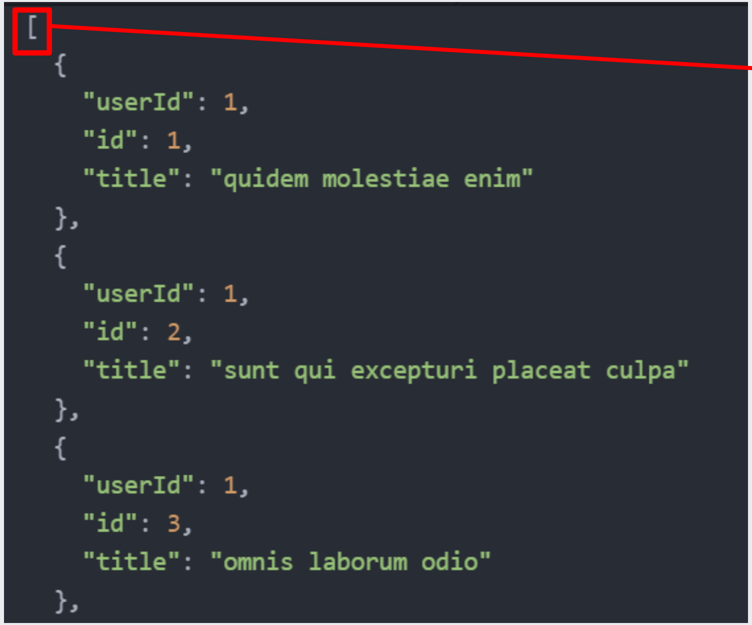
https://jsonplaceholder.typicode.com/albums

Kettle Implementation



```
2022/02/10 18:13:39 - Spoon - Running transformation using the Kettle execution engine
2022/02/10 18:13:39 - Spoon - Transformation opened.
2022/02/10 18:13:39 - Spoon - Launching transformation [albums]...
2022/02/10 18:13:39 - Spoon - Started the transformation execution.
2022/02/10 18:13:39 - albums - Dispatching started for transformation [albums]
2022/02/10 18:13:39 - Albums Table output.0 - Connected to database [lab_db] (commit=1000)
2022/02/10 18:13:39 - Albums Generate rows.0 - Finished processing (I=0, O=0, R=0, W=1, U=0, E=0)
2022/02/10 18:13:39 - Albums REST client.0 - Finished processing (I=0, O=0, R=1, W=1, U=0, E=0)
2022/02/10 18:13:39 - Albums JSON input.0 - Finished processing (I=100, O=0, R=1, W=100, U=0, E=0)
2022/02/10 18:13:39 - Select values.0 - Finished processing (I=0, O=0, R=100, W=100, U=0, E=0)
2022/02/10 18:13:39 - Albums Dummy.0 - Finished processing (I=0, O=0, R=100, W=100, U=0, E=0)
2022/02/10 18:13:40 - Albums Table output.0 - Finished processing (I=0, O=100, R=100, W=100, U=0, E=0)
2022/02/10 18:13:40 - Spoon - The transformation has finished!!
2022/02/10 18:46:26 - Spoon - Running transformation using the Kettle execution engine
2022/02/10 18:46:26 - Spoon - Transformation opened.
2022/02/10 18:46:26 - Spoon - Launching transformation [photos]...
2022/02/10 18:46:26 - Spoon - Started the transformation execution.
2022/02/10 18:46:47 - Spoon - The transformation has finished!!
```

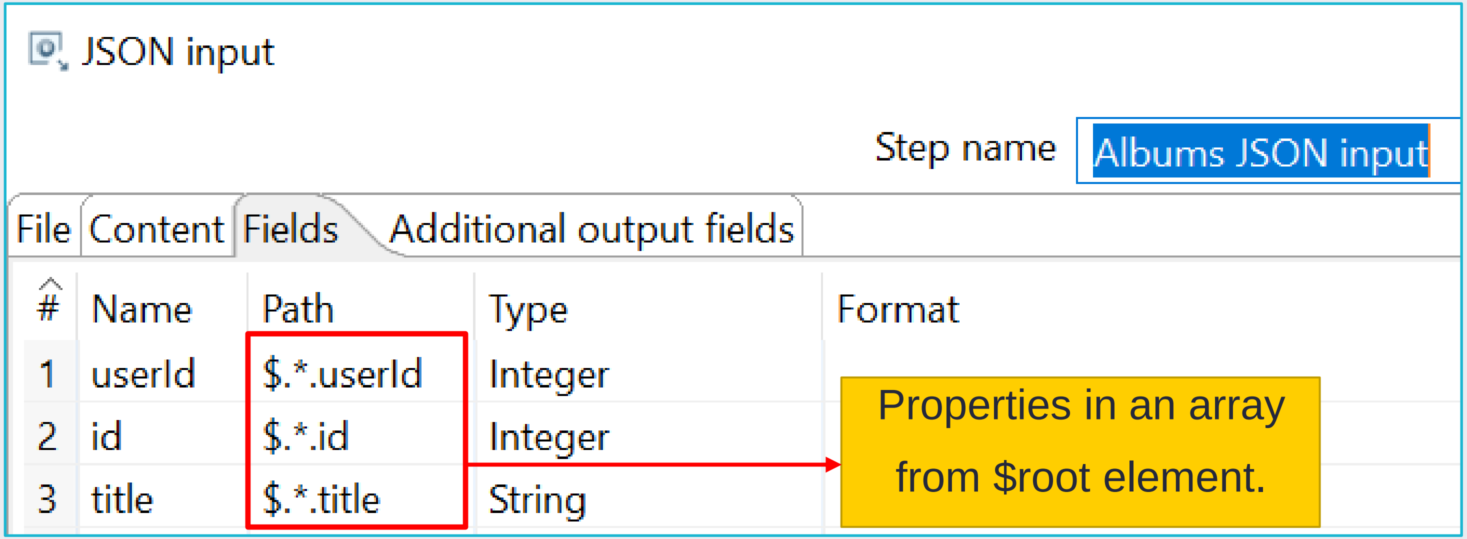
JSON Data: albums.json



```
[
  {
    "userId": 1,
    "id": 1,
    "title": "quidem molestiae enim"
  },
  {
    "userId": 1,
    "id": 2,
    "title": "sunt qui excepturi placeat culpa"
  },
  {
    "userId": 1,
    "id": 3,
    "title": "omnis laborum odio"
  },
```

The file is an array which is an ordered collection of values.

JSON Input



JSON input

Step name: Albums JSON input

#	Name	Path	Type	Format
1	userId	$.*.userId	Integer	
2	id	$.*.id	Integer	
3	title	$.*.title	String	

Properties in an array from $root element.

Table Output

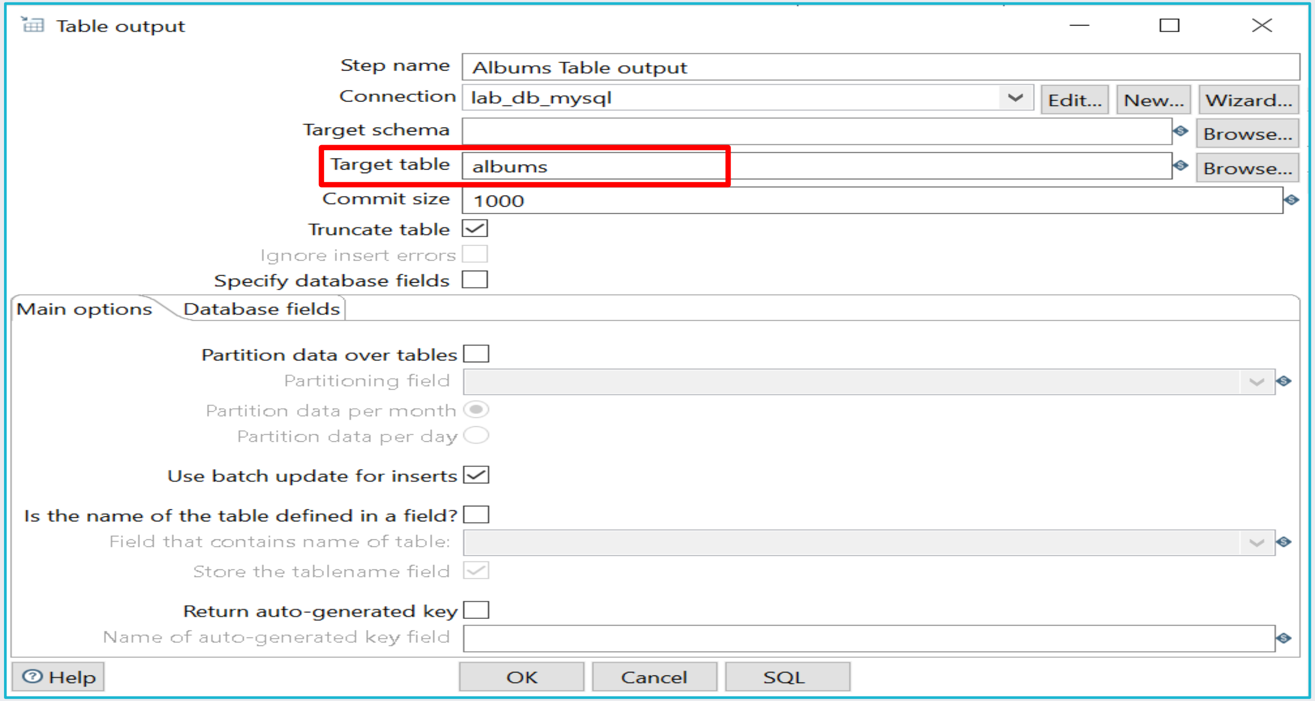


Table output

Step name: Albums Table output

Connection: lab_db_mysql

Target schema:

Target table: albums

Commit size: 1000

Truncate table ☑

Ignore insert errors □

Specify database fields □

Main options | Database fields

Partition data over tables □

Partitioning field:

Partition data per month ●

Partition data per day ○

Use batch update for inserts ☑

Is the name of the table defined in a field? □

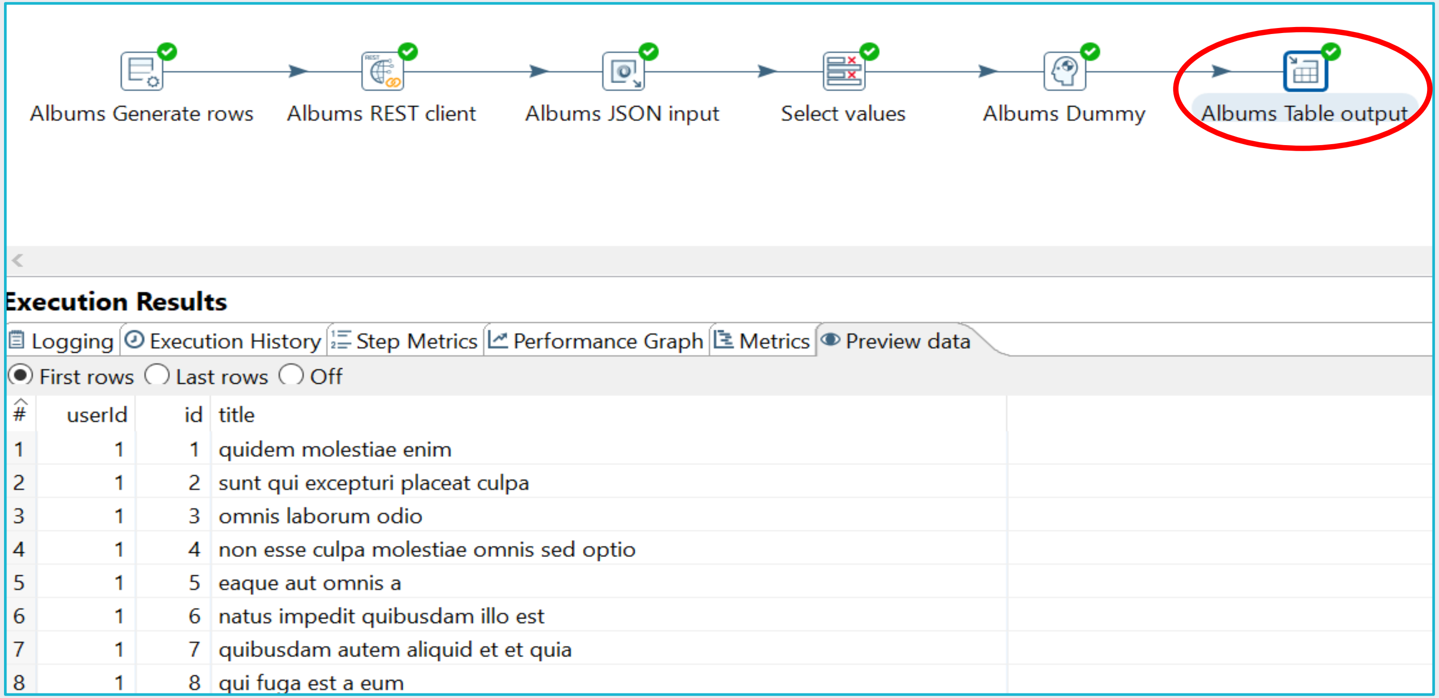
Field that contains name of table:

Store the tablename field ☑

Return auto-generated key □

Name of auto-generated key field:

Preview Data (Albums Table)

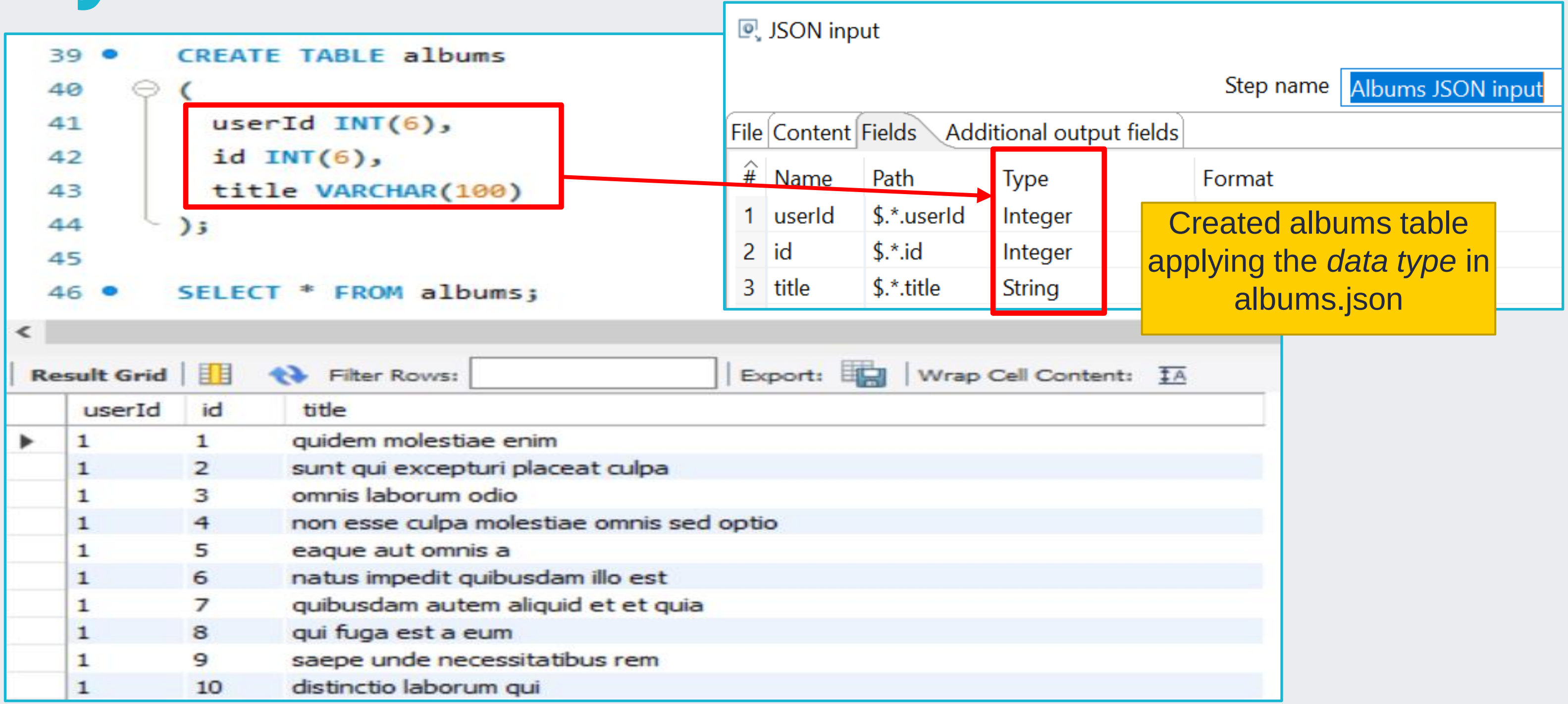


Execution Results

● First rows ○ Last rows ○ Off

#	userId	id	title
1	1	1	quidem molestiae enim
2	1	2	sunt qui excepturi placeat culpa
3	1	3	omnis laborum odio
4	1	4	non esse culpa molestiae omnis sed optio
5	1	5	eaque aut omnis a
6	1	6	natus impedit quibusdam illo est
7	1	7	quibusdam autem aliquid et et quia
8	1	8	qui fuga est a eum

MySQL Workbench: Albums



JSON input

Step name: Albums JSON input

File | Content | Fields | Additional output fields

#	Name	Path	Type	Format
1	userId	$.*.userId	Integer	
2	id	$.*.id	Integer	
3	title	$.*.title	String	

Created albums table applying the *data type* in albums.json

```
CREATE TABLE albums
(
    userId INT(6),
    id INT(6),
    title VARCHAR(100)
);

SELECT * FROM albums;
```

userId	id	title
1	1	quidem molestiae enim
1	2	sunt qui excepturi placeat culpa
1	3	omnis laborum odio
1	4	non esse culpa molestiae omnis sed optio
1	5	eaque aut omnis a
1	6	natus impedit quibusdam illo est
1	7	quibusdam autem aliquid et et quia
1	8	qui fuga est a eum
1	9	saepe unde necessitatibus rem
1	10	distinctio laborum qui

References

https://goessner.net/articles/JsonPath/

https://www.json.org/json-en.html

https://help.hitachivantara.com/Documentation/Pentaho/8.3/Products/Table_Output