

Variant Control Demo Outputs

Name: Dharal Naik

Reg No.: RA2311003010893

Different FastAPI Endpoints

POST	/Add_Product_Name/ Add Product	▼
POST	/Add_Product_and_Variants/ Add Product And Variants	▼
POST	/Add_Variants_for_Existing_Product/{product_name}/ Add Variants	▼
GET	/Get_Full_Data/{product_name} Get Product	▼
PUT	/Update_Product_Data/{product_id} Update Product	▼
PUT	/Update_Variant_Data/{variant_id} Update Variant	▼
DELETE	/Delete_Variant/{variant_id} Delete Variant	▼
DELETE	/Delete_all_Variants/{product_name} Delete All Variants	▼
DELETE	/Delete_Product/{product_name} Delete Product	▼

Total 9 Endpoints, 3 to Create data, 1 to Read data, 2 to Update data, 3 to Delete Data.

Create Endpoints

- There are three different endpoints to post and store data in a database using FastAPI.
- They are :-
 1. First to just add a product name to the database.
 2. Second to add both product name and its variants to the database.
 3. Third to add only variants of the pre-existing products in the database.

POST	/Add_Product_Name/	Add Product
POST	/Add_Product_and_Variants/	Add Product And Variants
POST	/Add_Variants_for_Existing_Product/{product_name}/	Add Variants

1. Add Product Name

POST Send

Query Headers 2 Auth Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {  
2   "product_name": "Tshirts"  
3 }
```

Status: 200 OK Size: 46 Bytes Time: 52 ms

Response Headers 5 Cookies Results Docs

```
1 {  
2   "Message": "Product Name added successfully."  
3 }
```

The above picture is the thunder client output for POST request.
The below two pictures are the changes made to database after this query.

1 select * from products

Data Output Messages Notifications

	pduct_id [PK] integer	pduct_name character varying
1	1	Tshirts

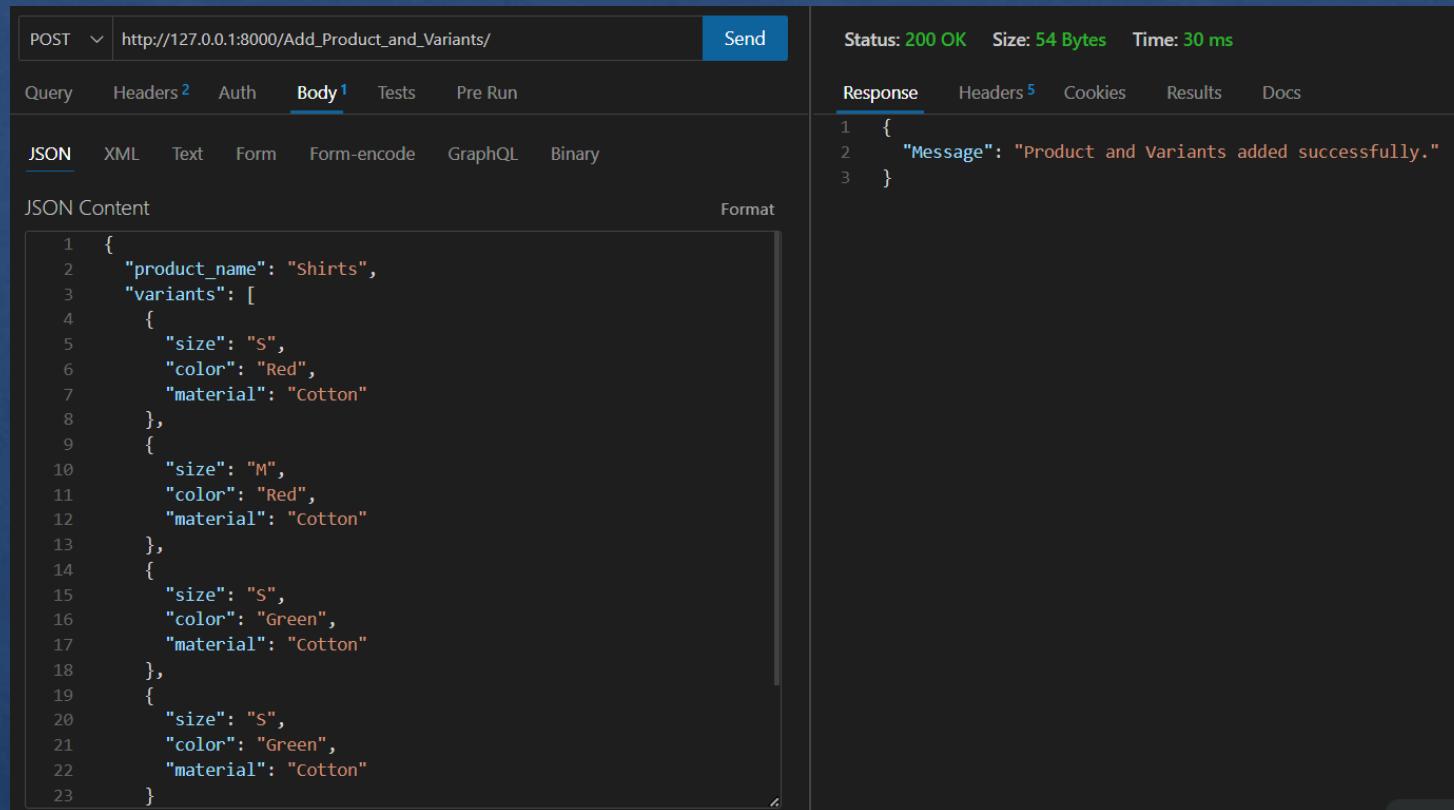
Query Query History

1 select * from variants

Data Output Messages Notifications

	var_id [PK] integer	size character varying	color character varying	material character varying	product_id integer	product_name character varying
1						

2. Add Product and Variants



The screenshot shows the Thunder Client interface with a POST request to `http://127.0.0.1:8000/Add_Product_and_Variants/`. The request body is a JSON object containing a product name and four variants. The response is a 200 OK status with a message indicating success.

Request Body:

```
1 {
2   "product_name": "Shirts",
3   "variants": [
4     {
5       "size": "S",
6       "color": "Red",
7       "material": "cotton"
8     },
9     {
10       "size": "M",
11       "color": "Red",
12       "material": "cotton"
13     },
14     {
15       "size": "S",
16       "color": "Green",
17       "material": "Cotton"
18     },
19     {
20       "size": "S",
21       "color": "Green",
22       "material": "Cotton"
23     }
]
```

Response:

```
1 {
2   "Message": "Product and Variants added successfully."
3 }
```

The above picture is the thunder client output for POST request.

3. Add Variants for Existing Product

The screenshot shows a POST request to `http://127.0.0.1:8000/Add_Variants_for_Existing_Product/Blazers/`. The request body is a JSON array of two objects, each representing a variant with size, color, and material. The response status is 200 OK, size is 42 Bytes, and time is 18 ms. The response body is a JSON object with a single key "Message" containing the value "Variants added successfully."

```
POST http://127.0.0.1:8000/Add_Variants_for_Existing_Product/Blazers/
Status: 200 OK Size: 42 Bytes Time: 18 ms

Response Headers Cookies Results Docs
1 {
2   "Message": "Variants added successfully."
3 }
```

```
[{"size": "XL", "color": "Grey", "material": "Silk"}, {"size": "L", "color": "Black", "material": "Cotton"}]
```

The screenshot shows a POST request to `http://127.0.0.1:8000/Add_Variants_for_Existing_Product/Caps/`. The request body is a JSON array of one object, representing a variant with size, color, and material. The response status is 404 Not Found, size is 40 Bytes, and time is 11 ms. The response body is a JSON object with a single key "detail" containing the value "Message: Product not found."

```
POST http://127.0.0.1:8000/Add_Variants_for_Existing_Product/Caps/
Status: 404 Not Found Size: 40 Bytes Time: 11 ms

Response Headers Cookies Results Docs
1 {
2   "detail": "Message: Product not found."
3 }
```

```
[{"size": "M", "color": "White", "material": "Cotton"}]
```

The first picture is the thunder client output for POST request, where variants are added to the product name – Blazers.

The second picture is to show the output when product is not found in the database, giving error 404 Not Found.

Database SS after all three POST requests

```
1 select * from products
```

Data Output Messages Notifications

	pduct_id [PK] integer	pduct_name character varying
1		Tshirts
2		Shirts
3		Blazers

```
1 select * from variants
```

Data Output Messages Notifications

	var_id [PK] integer	size character varying	color character varying	material character varying	product_id integer	product_name character varying
1	1	S	Red	Cotton	2	Shirts
2	2	M	Red	Cotton	2	Shirts
3	3	S	Green	Cotton	2	Shirts
4	4	M	Green	Cotton	2	Shirts
5	5	XL	Grey	Silk	3	Blazers
6	6	L	Black	Cotton	3	Blazers

Read Endpoints

- There is one endpoint to read the data from the database using FastAPI.
- Here, user gives input as the product name and gets the output of all of its variants.

```
GET /Get_Full_Data/{product_name} Get Product
```

The screenshot shows a Postman request for the endpoint `http://127.0.0.1:8000/Get_Full_Data/Shirts/`. The 'Query' tab is selected. The response status is **200 OK**, size is **229 Bytes**, and time is **11 ms**. The response body is a JSON array of four objects, each representing a shirt variant:

```
1 {  
2   "Product-": "Shirts",  
3   "Variants-": [  
4     {  
5       "Size": "S",  
6       "Color": "Red",  
7       "Material": "Cotton"  
8     },  
9     {  
10      "Size": "M",  
11      "Color": "Red",  
12      "Material": "Cotton"  
13    },  
14    {  
15      "Size": "S",  
16      "Color": "Green",  
17      "Material": "Cotton"  
18  },  
19  {  
20      "Size": "M",  
21      "Color": "Green",  
22      "Material": "Cotton"  
23    }]
```

The screenshot shows a Postman request for the endpoint `http://127.0.0.1:8000/Get_Full_Data/Tshirts/`. The 'Query' tab is selected. The response status is **200 OK**, size is **77 Bytes**, and time is **6 ms**. The response body is a JSON object with a single key-value pair:

```
1 {  
2   "Product-": "Tshirts",  
3   "Variants-": "No variants are added to the database."  
4 }
```

First picture shows output for product – Shirts, which has 4 variants in the database.
Second picture shows output for product – Tshirts, which has no variants in the database.

Update Endpoints

- There are two endpoints to update the data from the database using FastAPI, one to update Product Data and other to update Variants Data.
- Here, user gives input as the product id and variant id, which allows it to update the respective data.

PUT	/Update_Product_Data/{product_id}	Update Product
PUT	/Update_Variant_Data/{variant_id}	Update Variant

1. Update Product Data

The screenshot shows a Thunder Client interface for a PUT request to `http://127.0.0.1:8000/Update_Product_Data/3`. The 'Body' tab is selected, containing the following JSON payload:

```
1 {
2   "product_update": {
3     "product_name": "Suits"
4   },
5   "variant_update": {
6     "product_name": "Suits"
7   }
8 }
```

The response status is **200 OK**, size is **52 Bytes**, and time is **16 ms**. The response body is:

```
1 {
2   "Message": "Product data was updated successfully."
3 }
```

The screenshot shows a Thunder Client interface for a PUT request to `http://127.0.0.1:8000/Update_Product_Data/1`. The 'Body' tab is selected, containing the following JSON payload:

```
1 {
2   "product_update": {
3     "product_name": "Tee"
4   },
5   "variant_update": {
6     "product_name": "Tee"
7   }
8 }
```

The response status is **200 OK**, size is **50 Bytes**, and time is **8 ms**. The response body is:

```
1 {
2   "Message": "No variant data found, add variants."
3 }
```

The first picture is the thunder client output for PUT request, where product – Blazers is updated to Suits. The second picture is to show the output when product – Tshirts is not found in the database for variants.

2. Update Variant Data

The screenshot shows a Thunder Client interface for a PUT request to `http://127.0.0.1:8000/Update_Variant_Data/6/`. The 'Body' tab is selected, showing JSON content:

```
1 {  
2   "size": "XS",  
3   "color": "Off-White",  
4   "material": "Terry Cotton"  
5 }
```

The response status is **200 OK**, size is **52 Bytes**, and time is **10 ms**. The response body is:

```
1 {  
2   "Message": "Variant data was updated successfully."  
3 }
```

The screenshot shows a Thunder Client interface for a PUT request to `http://127.0.0.1:8000/Update_Variant_Data/7/`. The 'Body' tab is selected, showing JSON content:

```
1 {  
2   "size": "XS",  
3   "color": "Off-White",  
4   "material": "Terry Cotton"  
5 }
```

The response status is **404 Not Found**, size is **40 Bytes**, and time is **8 ms**. The response body is:

```
1 {  
2   "detail": "Message: Variant not found."  
3 }
```

The first picture is the thunder client output for PUT request, where variant data is updated.

The second picture is to show the output when variant is not found in the database, and its same for product data.

Database SS after both PUT requests

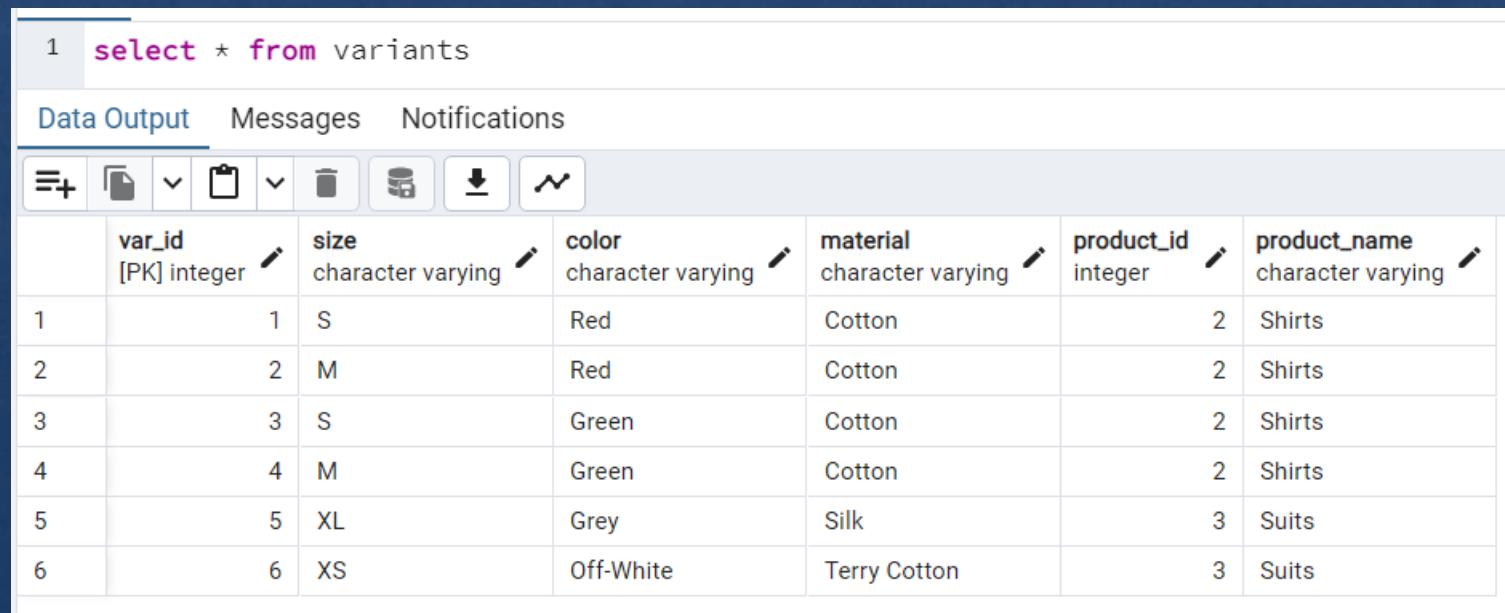
```
1 select * from products
```

Data Output Messages Notifications

	product_id [PK] integer	product_name character varying
1		Tshirts
2		Shirts
3		Suits

```
1 select * from variants
```

Data Output Messages Notifications



	var_id [PK] integer	size character varying	color character varying	material character varying	product_id integer	product_name character varying
1		S	Red	Cotton	2	Shirts
2		M	Red	Cotton	2	Shirts
3		S	Green	Cotton	2	Shirts
4		M	Green	Cotton	2	Shirts
5		XL	Grey	Silk	3	Suits
6		XS	Off-White	Terry Cotton	3	Suits

Delete Endpoints

- There are three endpoints to delete the data from the database using FastAPI.
- They are :-
 1. First to just delete one variant from the database.
 2. Second to delete all the variants from the database, to allow to delete product without foreign key issues.
 3. Third to delete the product from the database.

DELETE	<code>/Delete_Variant/{variant_id}</code>	Delete Variant
DELETE	<code>/Delete_all_Variants/{product_name}</code>	Delete All Variants
DELETE	<code>/Delete_Product/{product_name}</code>	Delete Product

1. Delete one Variant

The screenshot shows the Thunder Client interface. The top bar has 'DELETE' selected, the URL is 'http://127.0.0.1:8000/Delete_Variant/6', and the 'Send' button is highlighted. Below the URL, there are tabs for 'Query', 'Headers 2', 'Auth', 'Body 1', 'Tests', and 'Pre Run'. The 'Query' tab is active, showing 'Query Parameters' with a table for adding parameters. The 'Response' tab is also active, displaying the following JSON output:

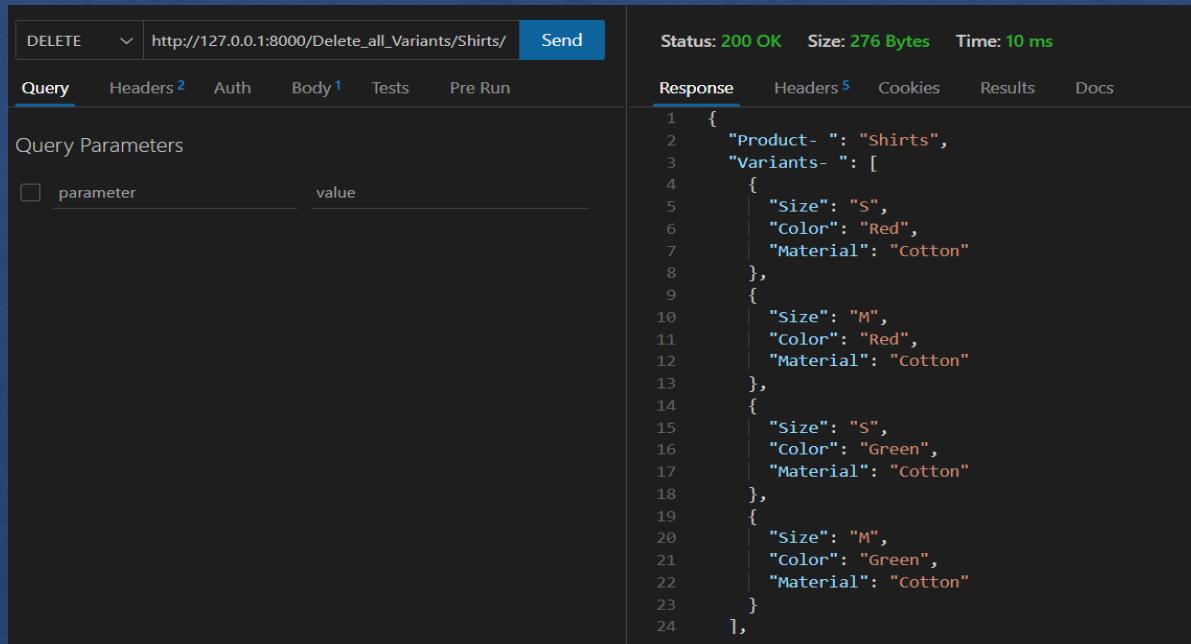
```
1 {
2   "Product-": "Suits",
3   "Variant-": {
4     "Size": "XL",
5     "Colour": "Off-White",
6     "Material": "Terry Cotton"
7   },
8   "Message": "Variant deleted successfully."
9 }
```

The above picture is the thunder client output for DELETE request.
The below picture is the changes made to database after deleting one variants of Suits.

The screenshot shows the DBeaver interface with a SQL query window containing the command 'select * from variants'. Below the query window is a table titled 'Data Output' with columns: var_id [PK] integer, size character varying, color character varying, material character varying, product_id integer, and product_name character varying. The table displays the following data:

	var_id [PK] integer	size character varying	color character varying	material character varying	product_id integer	product_name character varying
1	1	S	Red	Cotton	2	Shirts
2	2	M	Red	Cotton	2	Shirts
3	3	S	Green	Cotton	2	Shirts
4	4	M	Green	Cotton	2	Shirts
5	5	XL	Grey	Silk	3	Suits

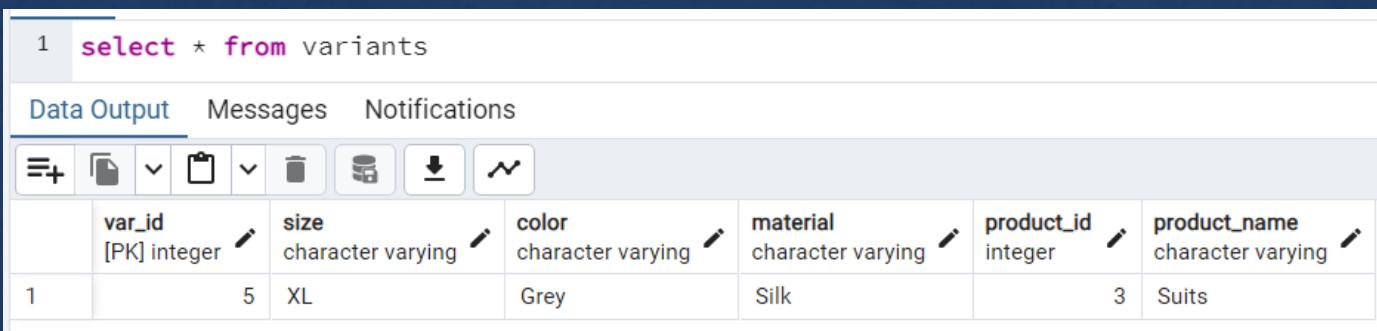
2. Delete all Variants



The screenshot shows the Thunder Client interface. The top bar has 'DELETE' selected and the URL 'http://127.0.0.1:8000/Delete_all_Variants/Shirts/'. Below the URL are tabs for 'Query', 'Headers', 'Auth', 'Body', 'Tests', and 'Pre Run'. The 'Query' tab is active, showing 'Query Parameters' with a table for adding parameters. The 'Response' tab shows the server's response:

```
1 {
2   "Product": "Shirts",
3   "Variants": [
4     {
5       "size": "S",
6       "color": "Red",
7       "Material": "Cotton"
8     },
9     {
10       "size": "M",
11       "color": "Red",
12       "Material": "Cotton"
13     },
14     {
15       "size": "S",
16       "color": "Green",
17       "Material": "Cotton"
18     },
19     {
20       "size": "M",
21       "color": "Green",
22       "Material": "Cotton"
23     }
24 ]
```

The above picture is the thunder client output for DELETE request.
The below picture is the changes made to database after deleting all variants for Shirts.



The screenshot shows the DBeaver interface with a SQL query window containing the command 'select * from variants'. Below the query are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with the following data:

	var_id [PK] integer	size character varying	color character varying	material character varying	product_id integer	product_name character varying
1	5	XL	Grey	Silk	3	Suits

3. Delete Product

The screenshot shows a DELETE request to `http://127.0.0.1:8000/Delete_Product/Shirts/`. The response status is 200 OK, size is 64 Bytes, and time is 10 ms. The response body is a JSON object: { "Product-": "Shirts", "Message": "Product deleted successfully." }.

DELETE	▼	http://127.0.0.1:8000/Delete_Product/Shirts/	Send
Query Headers 2 Auth Body 1 Tests Pre Run			
Query Parameters			
<input type="checkbox"/> parameter	value		

Status: 200 OK	Size: 64 Bytes	Time: 10 ms
Response Headers 5 Cookies Results Docs		
{ "Product-": "Shirts", "Message": "Product deleted successfully." }		

The screenshot shows a DELETE request to `http://127.0.0.1:8000/Delete_Product/Suits/`. The response status is 200 OK, size is 92 Bytes, and time is 5 ms. The response body is a JSON object: { "Message": "Cannot delete due to linked data. First go to delete variants and then repeat." }.

DELETE	▼	http://127.0.0.1:8000/Delete_Product/Suits/	Send
Query Headers 2 Auth Body 1 Tests Pre Run			
Query Parameters			
<input type="checkbox"/> parameter	value		

Status: 200 OK	Size: 92 Bytes	Time: 5 ms
Response Headers 5 Cookies Results Docs {}		
{ "Message": "Cannot delete due to linked data. First go to delete variants and then repeat." }		

The above two pictures are the thunder client output for DELETE request. The first picture is to show deletion of product – Shirts, and second is to show the reason behind no deletion of product – Suits.

3. Delete Product (Continued)

The screenshot shows the Thunder Client interface. The top bar has 'DELETE' selected, a URL of 'http://127.0.0.1:8000/Delete_Product/Shirts/', and a 'Send' button. Below the URL are tabs for 'Query', 'Headers 2', 'Auth', 'Body 1', 'Tests', and 'Pre Run'. The 'Query' tab is active. Under 'Query Parameters', there is a table with one row: a checkbox labeled 'parameter' and a text input labeled 'value'. The right panel shows the response: 'Status: 200 OK', 'Size: 64 Bytes', 'Time: 10 ms', and a 'Response' tab. The response body is a JSON object:

```
1 {  
2   "Product-": "Shirts",  
3   "Message": "Product deleted successfully."  
4 }
```

The above two pictures are the thunder client output for DELETE request.
The below picture shows the changes made to database, after deletion of product – Shirts.

The screenshot shows the pgAdmin interface with a query editor containing the SQL command 'select * from products'. Below the editor is a 'Data Output' tab. The table 'products' has two columns: 'product_id' [PK] integer and 'product_name' character varying. The data shows two rows: one with product_id 1 and product_name 'Tshirts', and another with product_id 2 and product_name 'Suits'.

	product_id	product_name
1	1	Tshirts
2	3	Suits

Thank you!