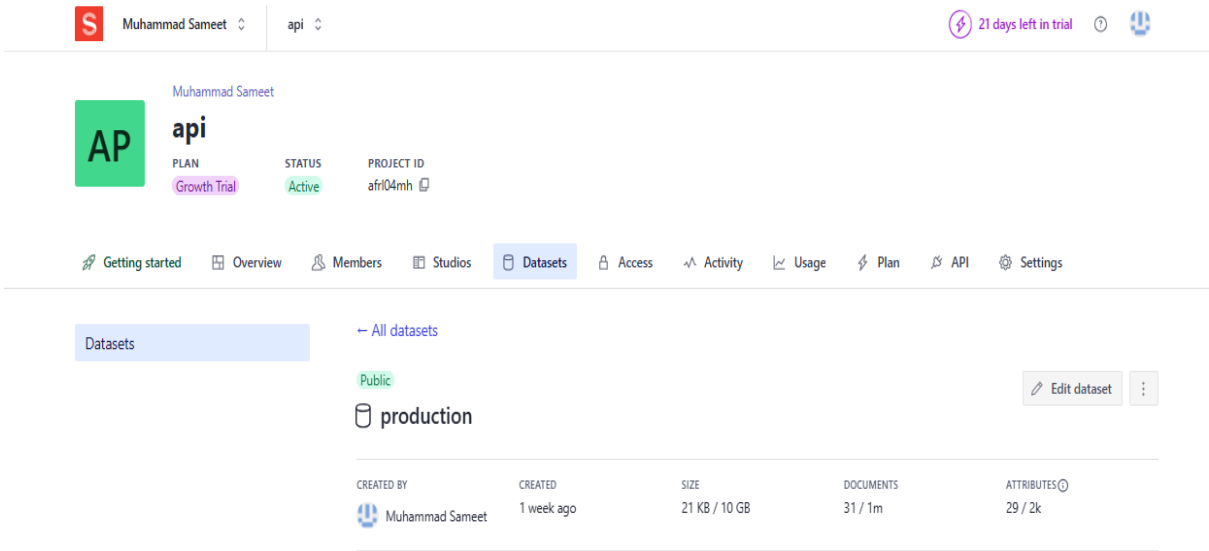
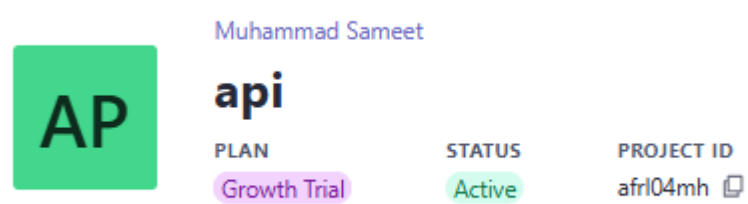


DAY 3 API Integration and Data Migration

1.Create sanity account project



1.Copy Id and add project Id yo.env file






* Open Click API Section

2.Add cores and origins

CORS origins

+ Add CORS origin

Hosts that can connect to the project API.

ORIGIN	CREDENTIALS	CREATED	
http://localhost:3002	Allowed	1 week	
http://localhost:3001	Allowed	1 week	
http://localhost:3000	Allowed	1 week	
...			

3. Create and add API Token to .env File

Name

Examples: "Employee import", "Website preview" or "PDF generator".

api

Permissions

Choose the access privileges for the token.

- ☐ Contributor
Read and write access to draft content within all datasets, with no access to project settings. (Tokens: read+write drafts)
- ☐ Deploy Studio (Token only)
Access to deploy Sanity Studio and GraphQL APIs to our hosted service.
- ☒ Developer
Read and write access to all datasets, with access to project settings for developers. (Tokens: read+write)
- ☐ Editor
Read and write access to all datasets, with limited access to project settings. (Tokens: read+write)
- ☐ Viewer
Read access to all datasets, with limited access to project settings. (Tokens: read-only)

Save

Cancel

Tokens

+ Add API token

4. Create Nextjs Project

Open and Run in terminal

```
npx create-next-app@latest
```

```
What is your project named? Api
you like to use TypeScript? Yes
Would you like to use ESLint? Yes
Would you like to use Tailwind CSS? Yes
Would you like your code inside a `src/` directory? Yes
Would you like to use App Router? (recommended) Yes
What import alias would you like configured? @/* add
```

After nextjs installation complete

5. Install Sanity In Nextjs Project Terminal

```
npm create sanity@latest
```

Add your sanity project name **API**

What is your project named? **Api**

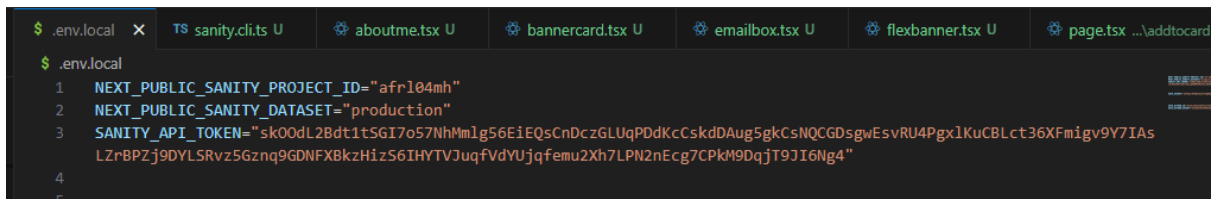
you like to use TypeScript? **Yes**

What route do you want to use for the studio? **/studio**

Select project template to use : **clean project with no predefined schema types**

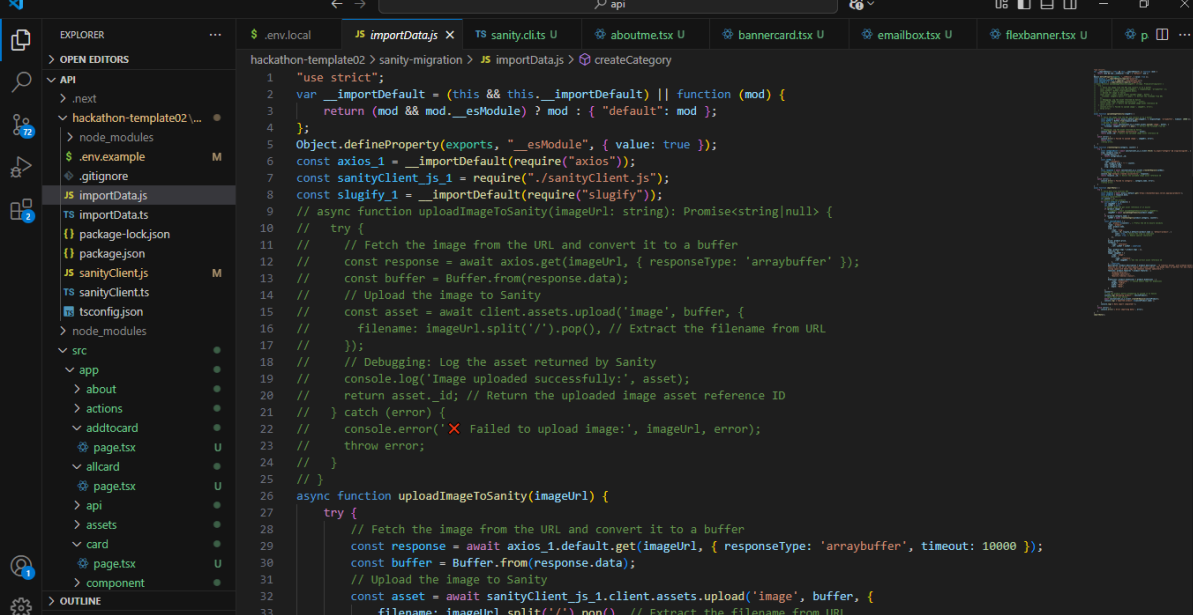
Would you like to add the project ID and dataset to your .env file?
Yes

.Env File of project



```
$ .env.local x TS sanity.cli.ts U aboutme.tsx U bannercard.tsx U emailbox.tsx U flexbanner.tsx U page.tsx ...addtocard
$ .env.local
1 NEXT_PUBLIC_SANITY_PROJECT_ID="afr104mh"
2 NEXT_PUBLIC_SANITY_DATASET="production"
3 SANITY_API_TOKEN="sk00dL2Bdt1tSGI7o57NhMmIg56EiEQsCnDczGLUqPDDKcCskdDAug5gkCsNQCGDsgwEsvRU4Pgx1KuCBLct36XFmigv9V7IAs
  LZrBPZj9DYL5Rvz5Gznq9GDNFXBkzHizS6IHYTVJuqfVdYUjqfemu2Xh7LPN2nEcg7CPkM9DqjT9JI6Ng4"
4
5
```

6. Create file in project ImportData.js

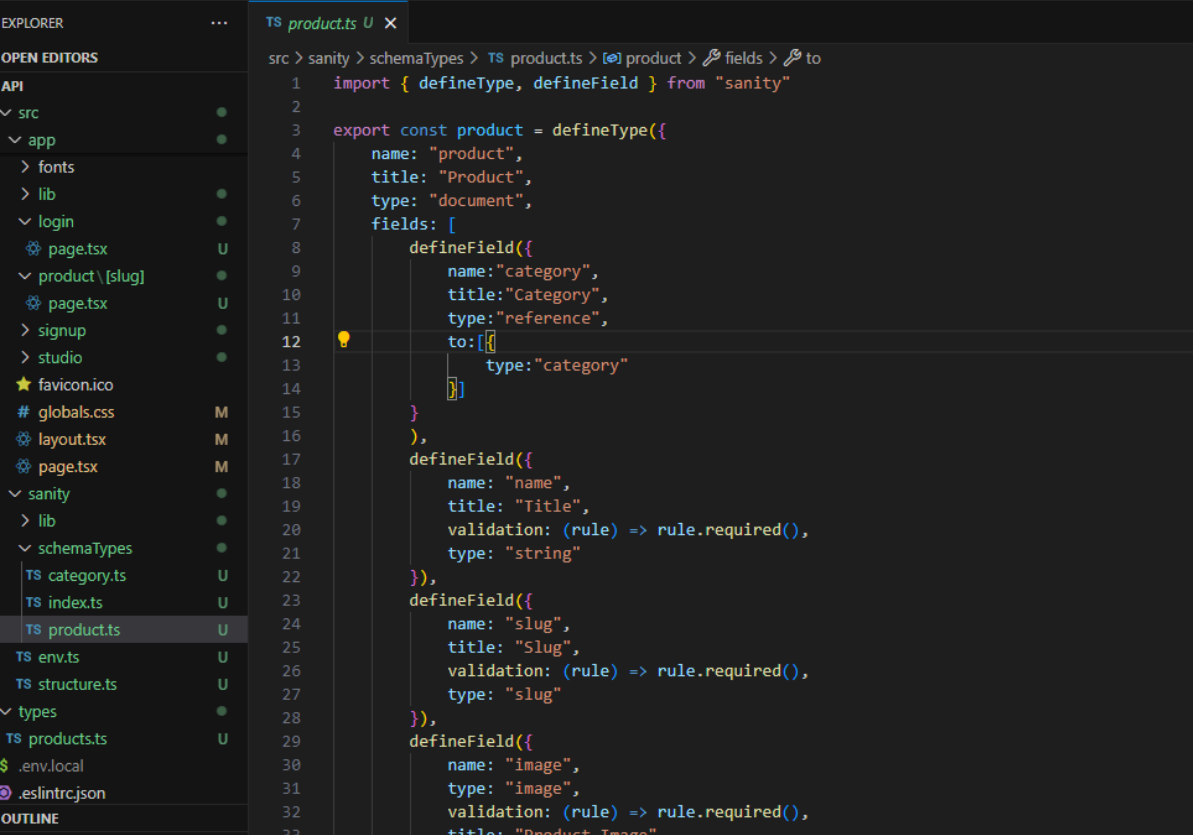


The screenshot shows a VS Code editor with the Explorer sidebar on the left. The file explorer shows a project structure with folders like 'API', 'src', and 'sanity'. The file 'importData.js' is highlighted in the Explorer. The main editor area shows the content of 'importData.js', which includes imports for 'sanityClient' and 'slugify', and a function 'uploadImageToSanity' that fetches an image from a URL and uploads it to Sanity.

```
1 "use strict";
2 var __importDefault = (this && this.__importDefault) || function (mod) {
3   return (mod && mod.__esModule) ? mod : { "default": mod };
4 };
5 Object.defineProperty(exports, "__esModule", { value: true });
6 const axios_1 = __importDefault(require("axios"));
7 const sanityClient_js_1 = require("./sanityClient.js");
8 const slugify_1 = __importDefault(require("slugify"));
9 // async function uploadImageToSanity(imageUrl: string): Promise<string|null> {
10 //   try {
11 //     // Fetch the image from the URL and convert it to a buffer
12 //     const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
13 //     const buffer = Buffer.from(response.data);
14 //     // Upload the image to Sanity
15 //     const asset = await client.assets.upload('image', buffer, {
16 //       filename: imageUrl.split('/').pop(), // Extract the filename from URL
17 //     });
18 //     // Debugging: Log the asset returned by Sanity
19 //     console.log('Image uploaded successfully:', asset);
20 //     return asset._id; // Return the uploaded image asset reference ID
21 //   } catch (error) {
22 //     console.error('Failed to upload image:', imageUrl, error);
23 //     throw error;
24 //   }
25 // }
26 async function uploadImageToSanity(imageUrl) {
27   try {
28     // Fetch the image from the URL and convert it to a buffer
29     const response = await axios_1.default.get(imageUrl, { responseType: 'arraybuffer', timeout: 10000 });
30     const buffer = Buffer.from(response.data);
31     // Upload the image to Sanity
32     const asset = await sanityClient_js_1.client.assets.upload('image', buffer, {
33       filename: imageUrl.split('/').pop(), // Extract the filename from URL
34     });
35   }
36 }
```

7. Create Sanity Schema in SanityTypes folder

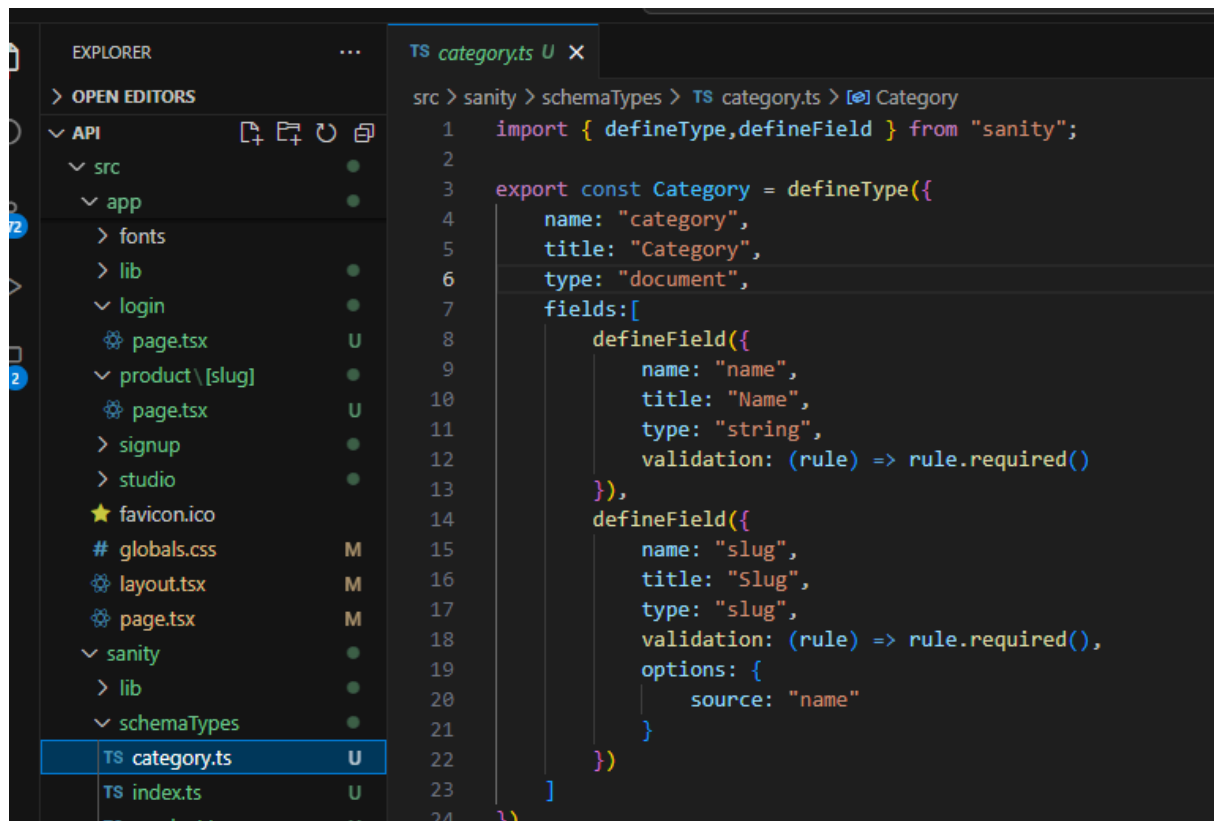
- product.ts



The screenshot shows a VS Code editor with the Explorer sidebar on the left. The file explorer shows a project structure with folders like 'src', 'sanity', and 'schemaTypes'. The file 'product.ts' is highlighted in the Explorer. The main editor area shows the content of 'product.ts', which includes imports for 'defineType' and 'defineField' from 'sanity', and a function 'product' that defines a Sanity schema for 'product' with fields 'category', 'name', 'slug', and 'image'.

```
1 import { defineType, defineField } from "sanity"
2
3 export const product = defineType({
4   name: "product",
5   title: "Product",
6   type: "document",
7   fields: [
8     defineField({
9       name: "category",
10      title: "Category",
11      type: "reference",
12      to: [
13        { type: "category" }
14      ]
15    }),
16    defineField({
17      name: "name",
18      title: "Title",
19      validation: (rule) => rule.required(),
20      type: "string"
21    }),
22    defineField({
23      name: "slug",
24      title: "Slug",
25      validation: (rule) => rule.required(),
26      type: "slug"
27    }),
28    defineField({
29      name: "image",
30      type: "image",
31      validation: (rule) => rule.required(),
32      title: "Product Image"
33    })
34 ]
35 })
```

- category.ts

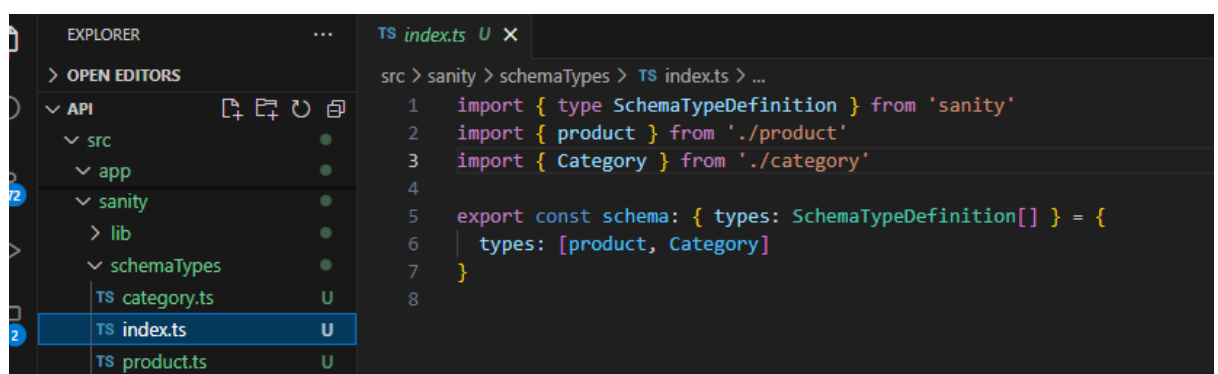


The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays the project structure with folders like 'src', 'app', 'lib', 'login', 'product', 'signup', 'studio', and 'sanity'. The 'sanity' folder is expanded, showing 'lib' and 'schemaTypes'. The 'category.ts' file is selected and highlighted. The main editor area shows the code for 'category.ts'.

```
src > sanity > schemaTypes > TS category.ts > [Category]
1  import { defineType, defineField } from "sanity";
2
3  export const Category = defineType({
4    name: "category",
5    title: "Category",
6    type: "document",
7    fields: [
8      defineField({
9        name: "name",
10       title: "Name",
11       type: "string",
12       validation: (rule) => rule.required()
13     }),
14     defineField({
15       name: "slug",
16       title: "Slug",
17       type: "slug",
18       validation: (rule) => rule.required(),
19       options: {
20         source: "name"
21       }
22     })
23   ]
24 })
```

8. Add schema files to SchemaTypes “schemas”

- Index.ts



The screenshot shows the VS Code interface. The Explorer sidebar is similar to the previous one, but 'category.ts' is no longer highlighted. The 'index.ts' file is selected and highlighted. The main editor area shows the code for 'index.ts'.

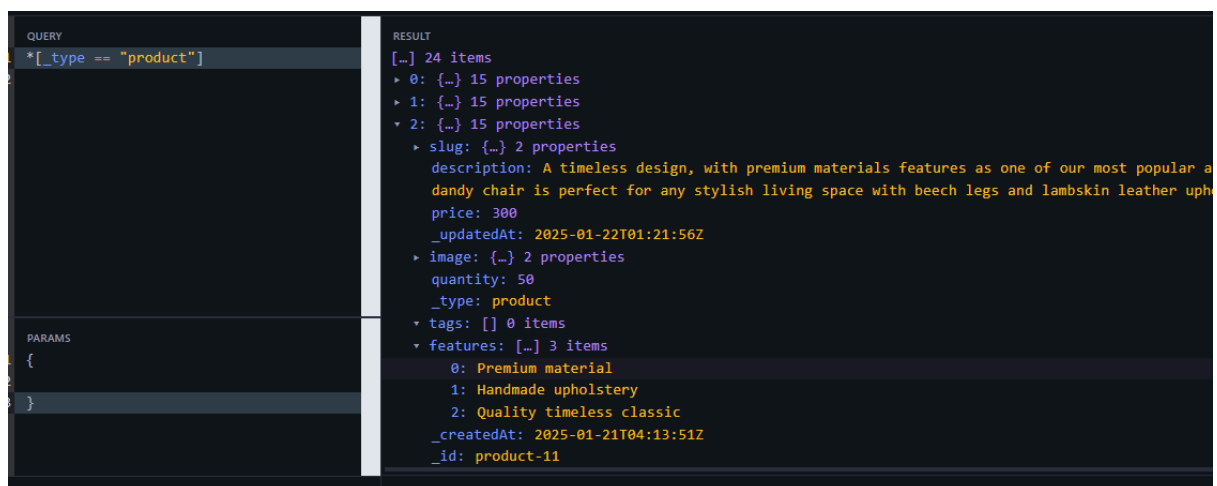
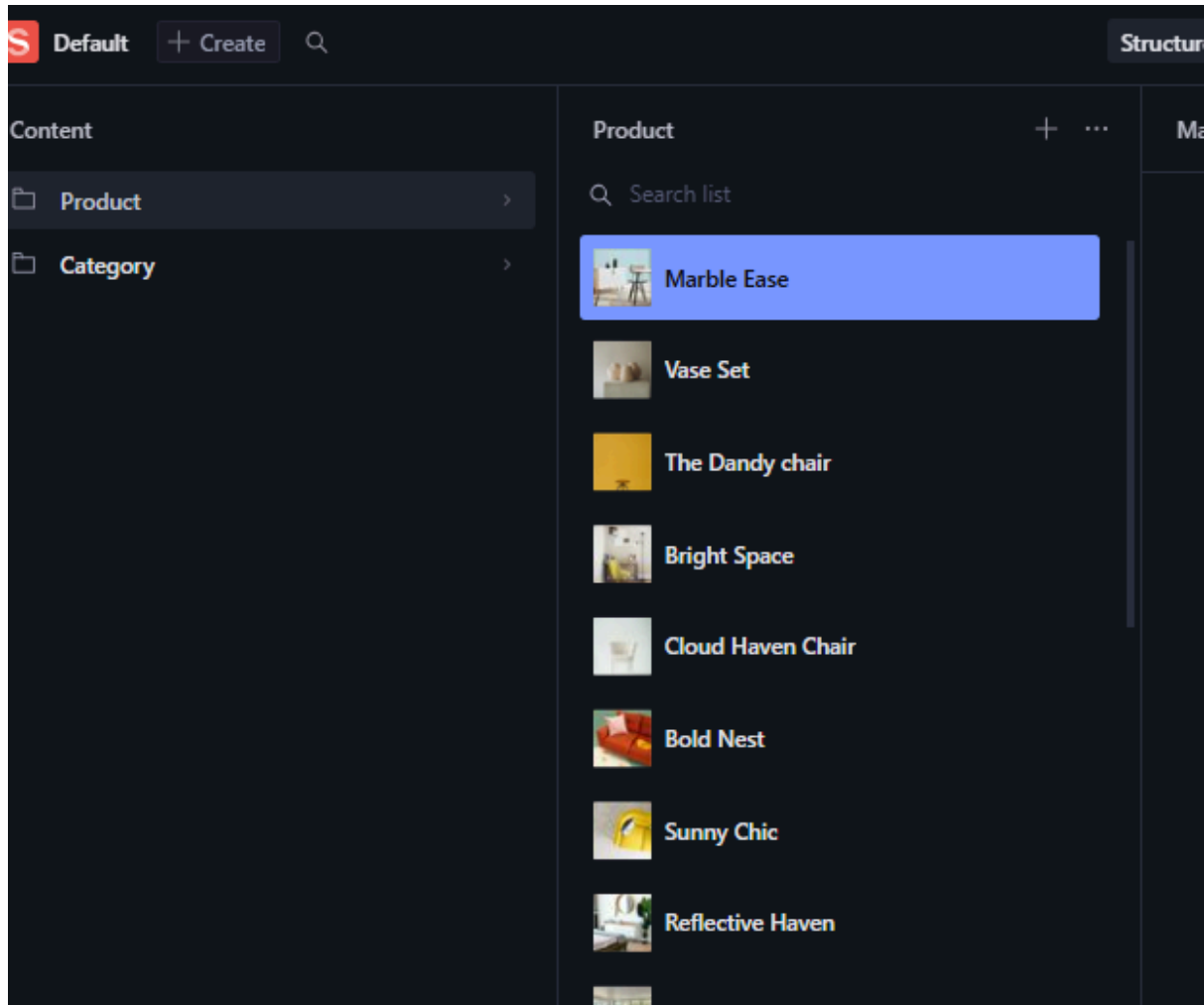
```
src > sanity > schemaTypes > TS index.ts > ...
1  import { type SchemaTypeDefinition } from 'sanity'
2  import { product } from './product'
3  import { Category } from './category'
4
5  export const schema: { types: SchemaTypeDefinition[] } = {
6    types: [product, Category]
7  }
8
```

9. Install in project terminal

- npm install axios
- Run Import-data

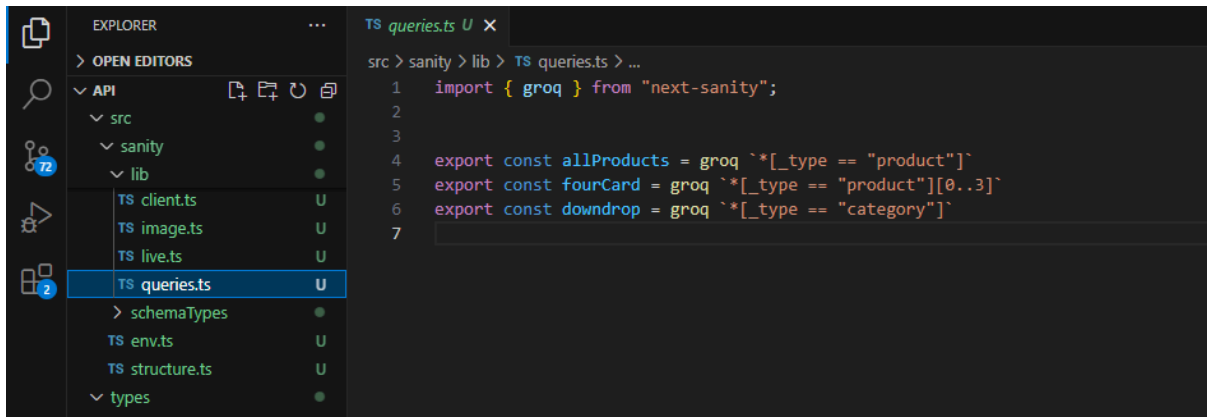
& import api data to sanity Project

10. API Data Import Successful To Sanity Project



11. Create Queries.ts File in Sanity folder

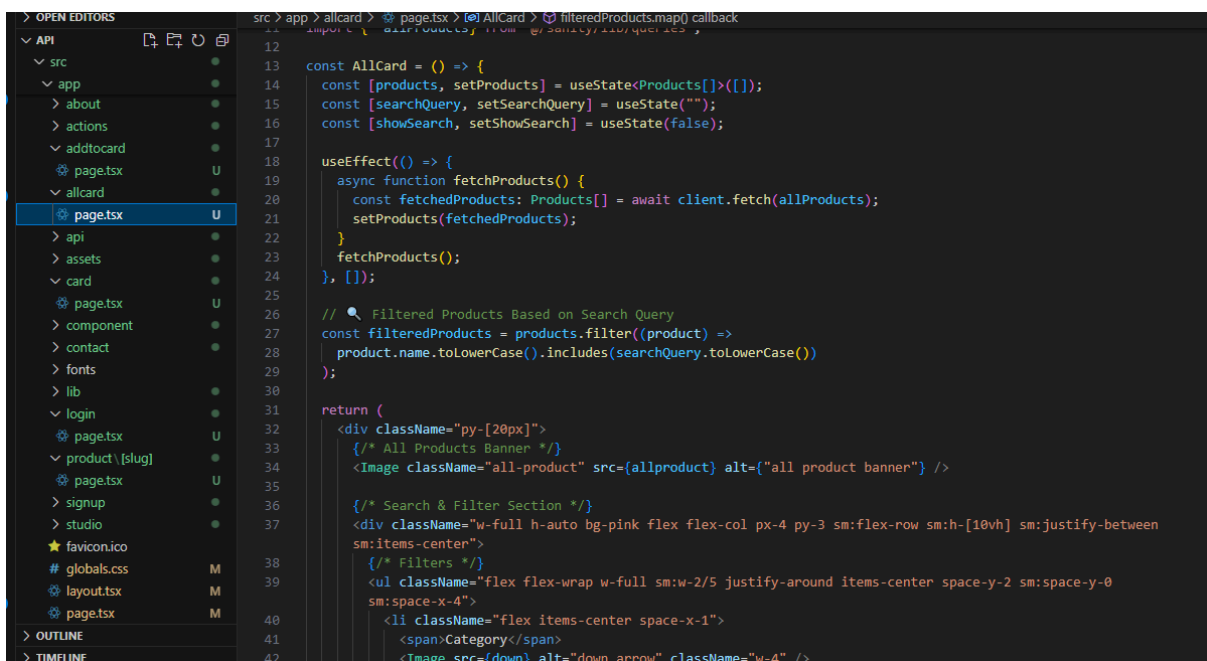
Create products & category “groq”



The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays the project structure: API > src > sanity > lib. The file 'TS queries.ts' is selected and highlighted. The main editor area shows the content of 'TS queries.ts' with the following code:

```
src > sanity > lib > TS queries.ts > ...
1  import { groq } from "next-sanity";
2
3
4  export const allProducts = groq `*[_type == "product"]`
5  export const fourCard = groq `*[_type == "product"][0..3]`
6  export const dropdown = groq `*[_type == "category"]`
7
```

12. Fetch all sanity data to product page



The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays the project structure: API > app > allcard > page.tsx. The file 'page.tsx' is selected and highlighted. The main editor area shows the content of 'page.tsx' with the following code:

```
src > app > allcard > page.tsx > [AllCard > filteredProducts.map() callback]
12
13 const AllCard = () => {
14   const [products, setProducts] = useState<Products[]>([]);
15   const [searchQuery, setSearchQuery] = useState("");
16   const [showSearch, setShowSearch] = useState(false);
17
18   useEffect(() => {
19     async function fetchProducts() {
20       const fetchedProducts: Products[] = await client.fetch(allProducts);
21       setProducts(fetchedProducts);
22     }
23     fetchProducts();
24   }, []);
25
26   // Filtered Products Based on Search Query
27   const filteredProducts = products.filter((product) =>
28     product.name.toLowerCase().includes(searchQuery.toLowerCase())
29   );
30
31   return (
32     <div className="py-[20px]">
33       <div className="flex flex-wrap w-full sm:w-2/5 justify-around items-center space-y-2 sm:space-y-0">
34         <div className="flex items-center space-x-1">
35           <span>Category</span>
36           <Image src={down} alt="down arrow" className="w-4" />
37         </div>
38       </div>
39     </div>
40   );
41
42
```


src

app

about

actions

addtocard

page.tsx

allcard

page.tsx

api

assets

card

page.tsx

component

contact

fonts

lib

login

page.tsx

product \ [slug]

page.tsx

signup

studio

favicon.ico

globals.css

layout.tsx

page.tsx

OUTLINE

TIMELINE

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

</div>

</div>

/* Products Section */

<div className="px-4 md:px-8 py-12 text-[#2A254B] mt-6">

<h1 className="text-2xl font-semibold">All Products</h1>

<div className="grid grid-cols-2 md:grid-cols-4 gap-8 mt-12">

{filteredProducts.length > 0 ? (

filteredProducts.map((product) => (

<div className="w-full h-auto key={product._id}>

<Link href={`/product/\${product.slug.current}`}>

{product.image && (

<Image

src={urlFor(product.image).url()}>

alt={product.name}

width={800}

height={800}

className="w-full h-[80%] object-cover transition-transform duration-300 ease-in-out

hover:scale-105 hover:translate-y-1">

</Image>

)

<div className="mt-4 text-[#2A254B]">

<p className="py-2">{product.name}</p>

<p>{product.price}</p>

</div>

</Link>

</div>

)

): (

<p className="text-center text-gray-500">No products found.</p>

)

</div>

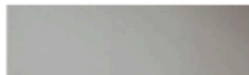
All Products



The Poplar suede sofa
980



Tropical Vibe
550



Sleek Living
300



Serene Seat
350

