MyEcommerceSite Apps -> An Ecommerce site, three SaaS internal business tools: MyEcommerceSite Employee App, MyEcommerceSite Manager App, MyEcommerceSite Developer App, with Postgres, Stripe API, and the Retool low-code platform, with order management dashboard, an employee dashboard, and a developer portal. The business tools are built with the Retool platform, which is a drag-and-drop no-code editor with many pre-built components to build internal CRUD (create, read, update, delete) apps.

Functionalities/Demo:

- An application used by employees, an application used by managers, and an application used by developers, all linked to one database for a small to medium-sized company.
- Useful internal business Apps to run an e-commerce store from the backend or office-side.
- The Apps will display all items in our inventory to sell, acquiring data from the database.
- Each item for sale has an image, a description, a URL, and a unique SQ or identifier for data management that allows us to manage the purchase made by customers.
- Searching for orders and issuing refunds based on Stripe API integration.
- Emailing customers with auto-generated texts with custom JavaScript as well as SMTP API.
- Viewing the general analytics of the Revenue Tracker.
- A B2B app which offers additional developer friendly functionalities, such as being able to facilitate orders made through API by displaying the endpoints.

Developing tools:

Retool
https://retool.com/?_keyword=retool&adgroupid=77096230789&utm_source=google&utm_medium=search&utm_campaign=6470119914&utm_term=retool&utm_content=595361152454&hsa_acc=7420316652&hsa_cam=6470119914&hsa_grp=77096230789&hsa_ad=595361152454&hsa_src=g&hsa_tgt=kwd-395242915847&hsa_kw=retool&hsa_mt=e&hsa_net=adwords&hsa_ver=3&gclid=Cj0KCQjwnbmaBhD-ARIsAGTPcfX8vZvorXMWz7qQCZeaEcR4kU76cuVnJFbkDYl-RHGT-ScmvwSPCNwaAlMdEALw_wcB
PostgreSQL Database https://github.com/kubowania/mobee-psql-data/blob/main/postgres.sql
Stripe https://stripe.com
Stripe Testing Documentation https://stripe.com/docs/terminal/references/testing
SMTP Email Relay Services https://www.smtp.com/
REST API Generator https://retool.com/api-generator/

Prerequisites & Setup:

Create an account on ReTool, along with creating an URL name for the workspace.
Create the App **MyEcommerceSite Employee App.** Delete the default query and table.
Create a new container that holds for employee's inventory tables.
Same setups for **MyEcommerceSite Manager App** and **MyEcommerceSite Developer App.**

Synchronous Developing Notes:

MyEcommerceSite Employee App:
Customize the essential containers and list views:
containers list views customized.PNG
Create a new resource query underneath and paste PostgreSQL in.
To manage sales order data, create a new resource query `getSalesOrders`:

```
SELECT * from salesorder;
```

Save and run, the sales order queries ran successfully:

query ran successfully.PNG

Retrieve sales data with `{{ getSalesOrders.data}}` in `orderTable`.

To get employees ID, create a new resource query `getEmployees`:

```
SELECT * from employee; {{getEmployees.data}}
```

Save and run, now all employees info data are fetched:

employees info data fetched.PNG

Create a new resource query for inventory named `getInventory`:

```
SELECT * from product;
```

Save and run, now all inventory data are fetched:

inventory data fetched.PNG

Replace the number of rows with `{{getInventory.data.productid.length}}`

Replace the product photo's URL with `{{getInventory.data.photo[i]}}`

Replace the product name with `{{getInventory.data.productname[i]}}`

Replace the price value with `{getInventory.data.unitprice[i]}}`

Replace the description with `{{ getInventory.data.productdescription[i] }}`

To get products info, create a new resource query `getProduct`:

```
SELECT * FROM product WHERE productid =
ANY({{getSalesOrderDetail.data.productid}})
```

product details displayed when click on order ID.PNG

Add customized site logo:

MySiteLogo.PNG

To fetch customer data, create a new resource query `getCustomer`:

```
SELECT * FROM customer WHERE custid = {{
OrderTable.selectedRow.data.custid}}
```

Create Refunds for certain customers with data from query:

```
Create Refund for {{getCustomer.data.contactname}}
```

Get the refund order details:

```
 
This is for order number {{OrderTable.selectedRow.data.orderid}} made
on {{new Date(OrderTable.selectedRow.data.orderdate).toDateString()}}
```

refund order details.PNG

Fetch to display the shipment data:

```
 This order was shipped on the {{new
Date(OrderTable.selectedRow.data.shippeddate).toDateString()}} to
{{OrderTable.selectedRow.data.shipaddress}}
```

shipment details fetched.PNG

We need a JavaScript Transformer to hold the email text:

```
const message = `Dear ${ {{ getCustomer.data.contactname[0]}} },
I am so sorry that your was ${ {{refundSelectInput.value}} },
Please rest assured that your refund of ${ {{currency1.value}} } USD
has been authorized and shall be with you in 1-5 business days.
Have a wonderful rest of the day!`
```

```
return message
```

Now the customized email composing session is completed: **email composition is done.PNG**

Obtain a fake charge ID by submitting a fake payment method on Stripe.

Then create a new Stripe resource on Retool with the Stripe test API key. Test Connection.

Create postRefund query based on Stripe resource, and enter the refunding amount:

refund email auto-generated.PNG

Configure SMTP with Microsoft Outlook since Gmaili prevents 3rd-party-access:

https://www.saleshandy.com/smtp/outlook-smtp-settings/

Then Authorize the outlook Protocol SMTP, add to `sendEmail` resource, add confetti effect:

Now SMTP POST authorization and confetti effect worked when click Send Email:

smtp send email works.PNG

And also a refund regarding order email shows:

refund regarding order email shows.PNG

MyEcommerceSite Admin App:

Customize the employee editing section:

initial customization of the admin employee.PNG

Create a new manage_db resource query `getEmployee` to fetch employee's data:

```
SELECT * from employee
```

fetched employee data.PNG

Get number of employees in list view:

```
{{getEmployees.data.empid.length}}
```

Loop to get the employee's photos:

```
{{getEmployees.data.photo[i]}}
```

Loop to get the first and last name of the employees:

```
{{getEmployees.data.firstname[i]}}
{{getEmployees.data.lastname[i]}}
```

all employee's data fetched.PNG

Create a new query `updateEmployee` to fetch the update button effect:

```
 UPDATE employee
SET firstname = {{textInput1.value}},
    lastname = {{textInput2.value}},
    title = {{textInput3.value}},
    birthdate = {{textInput4.value}},
    hiredate = {{textInput5.value}},
    address = {{textInput6.value}},
    city = {{textInput7.value}},
    region = {{textInput8.value}},
    postalcode = {{textInput9.value}},
    country = {{textInput10.value}},
    phone = {{textInput11.value}},
    empid = {{numberInput1.value}}
    WHERE empid = {{text14.value}}
```

Add some trigger confetti effect on the Update button, now if change Sara to Sarah:

update employee info successfully-1.PNG **update employee info successfully-2.PNG**

Create a new query `deleteEmployee` to delete employees:

```
DELETE FROM employee WHERE empid= {{text14.value}}
```

Create a new query `getInventory` to get inventories:

```
SELECT * FROM product
```

Create a new query `addProduct` to add products into the inventory list:

```
INSERT INTO product(productid, productname, productdescription,
unitprice, unitsinstock, photo)
VALUES({{numberInput2.value}}, {{textInput12.value}},
{{textArea1.value}},{{numberInput3.value}},{{currency1.value}},
{{textInput13.value}})
```

Add a testing Samsung phone product to see if product adding function works:

samsung phone test product added.PNG

Create a new query `deleteProduct` to delete products in Inventory list:

```
DELETE from product WHERE productid
{{InventoryTable.selectedRow.data.productid}}
```

Now delete the test product:

test product deleted.PNG

Revenue Tracker:

Create a new **Query JSON with SQL**:

```
SELECT * FROM {{formatDataAsArray(getOrderDetail.data)}} as detail
JOIN {{formatDataAsArray(getSalesOrders.data)}} as orders
ON orders.orderid = detail.orderid
```

Now the order id fetched:

order id fetched.PNG

Use `{{joinedRevenueData.data}}` to import revenue data.

revenue tracker final look-1.PNG

revenue tracker final look-2.PNG

MyEcommerceSite Developer App:

Map Data for sales order table:

```
 [{
    "Method Type": "GET",
    "Endpoint": ".../sales_orders/",
    "Action": "Copy"
  },
  {
    "Method Type": "GET filter",
    "Endpoint": ".../sales_orders?id=value",
    "Action": "Copy"
  },
  {
    "Method Type": "GET pagination",
```

```
    "Endpoint": ".../sales_orders?_page=2&limit10",
    "Action": "Copy"
  },
  {
    "Method Type": "POSTr",
    "Endpoint": ".../sales_orders",
    "Action": "Copy"
  },
  {
    "Method Type": "PUT",
    "Endpoint": ".../sales_orders/",
    "Action": "Copy"
  },
  {
    "Method Type": "DELETE",
    "Endpoint": ".../sales_orders/1",
    "Action": "copy"
  }]
```

Create `getSalesOrderData` query to fetch sales order data:

```
SELECT * from salesorder
```

Generate new sales_orders API from Rest API tools website on Retool.

The URLs of all actions can be copied:

urls of actions can be copied.PNG

Follow the same steps above to get Customers and Employees data:

Developer Portal sales order final look.PNG

Developer Portal customers final look.PNG

Developer Portal employees final look.PNG