

# The Lianja 11 AI Assistant

## [The Lianja 11 AI Assistant](#)

### [Executive Summary](#)

### [Build AI powered Apps for the future](#)

### [AI powered database management](#)

### [AI powered script editing](#)

### [Working in the Apps Workspace](#)

### [Working in the Data Workspace](#)

## [AI Assistant deep dive](#)

### [Pages AI Assistant](#)

### [Ask AI anything](#)

## [AI Conversations](#)

### [Conversation History](#)

### [Saving and Replaying Conversations](#)

### [Managing Conversations](#)

### [Email Conversations](#)

### [Understanding Conversation Context](#)

### [Suggestion Menus](#)

### [Editing Suggestion Menus](#)

## [Data AI Assistant](#)

### [AI Powered Business Intelligence](#)

### [Business Intelligence Reporting](#)

### [Data Visualization](#)

## [What can you do in the Data workspace](#)

## [Building custom AI Agents](#)

### [Let's build a custom AI Agent](#)

### [Installing the agent](#)

### [Testing the agent](#)

### [Publishing Agents into the Agents repository](#)

### [Agent discovery and installation](#)

### [Deploying agents into the Cloud](#)

## [Limitations and Restrictions](#)

## Executive Summary

**Lianja 11** is a comprehensive **AI Powered**, *data-centric* Application Platform as a Service (APaaS) designed to significantly boost developer productivity and streamline the entire lifecycle of business application development. The platform emphasizes visual design, rapid development, and flexible deployment across a multitude of platforms including Windows, Mac,

and Linux, as well as various client devices such as Desktop, Web, Mobile, Progressive Web Apps (PWA), and Electron Apps. A cornerstone of Lianja's value proposition is its unparalleled multi-language support, embedding JavaScript, LianjaScript (fully compatible with Visual FoxPro), Python and TypeScript. This unique integration allows developers to leverage their existing skillsets without the need to learn a new language, thereby minimizing retraining efforts and accelerating project timelines. This capability stands as a critical differentiator in a market where many platforms often restrict developers to a single language or a limited set of tools.

Lianja adopts a unique "Integrated **No-Code**, **Low-Code** and **Pro-Code** Tools" philosophy. This means it caters to a broad spectrum of users: from business domain experts (citizen developers) who can visually design and build applications with minimal or no coding using the intuitive **Lianja App Builder's** drag-and-drop interface, to professional developers who require the flexibility to extend functionality through code when needed. This hybrid approach allows for rapid prototyping and iterative refinement while ensuring deep customization capabilities.

Other notable features include a comprehensive set of UI classes that enable the creation of modern-looking applications, all of which can be easily customized and skinned using CSS, eliminating the need for hardcoding visual styles. A built-in, cross-platform web browser component (based on Webkit) and integrated server page processing engines for LianjaScript (.rsp), JavaScript (.jssp) and Python (.pysp) allow server-side logic within Lianja Apps without external installations. The **Lianja Cloud Server** embeds LianjaScript, the Google-developed V8 JavaScript engine and Python, providing robust support for server-side scripting. Advanced security features, including user roles and permissions (assignable at App, Page, Section, and Field levels, with static or dynamic application), Row Level Security (RLS), and Dynamic Data Masks (DDM) control data visibility and access. A built-in Data Dictionary for managing business rules (validation, permissions, calculated fields), Chronological Data Versioning (Timelines), and audit trails for regulatory compliance are also present. **Lianja SQL Server** is highly compatible with *Microsoft T-SQL* and is embedded into both the Lianja App Builder and the Lianja Cloud Server. Finally, the Lianja App Center serves as a secure, centralized hub for deploying and provisioning applications to end-users across desktop, web, and mobile devices, with customizable corporate branding.

## **Build AI powered Apps for the future**

In today's data-driven world, the ability to efficiently gather and process information is paramount for the success of artificial intelligence (AI) applications.

The next step in AI will be applications.

Lianja AI Assistant has full support for building custom AI agents that integrate seamlessly into your Lianja Apps.

# AI powered database management

The Lianja AI Assistant provides AI powered natural language queries in the Data Workspace of the App Builder and any page within an App. Desktop, Web and Mobile are all supported. Additionally and uniquely, Lianja 11 has T-SQL compatibility which has been extended with AI Natural Language Query capabilities (NLQ) using the PROMPT command. This new SQL command returns the result set from a SQL SELECT statement generated by AI when you ask it something in natural language.

The Lianja AI Assistant makes extensive use of T-SQL on the client side as it interacts with AI providers such as OpenAI.

The Lianja T-SQL database engine is embedded within the Lianja client at both development and runtime providing fast and reliable response times for even the most demanding Natural Language Queries asked by a user.

As you can see from the screenshot below, the AI Assistant is displayed on the right of the window.

The screenshot shows the Lianja IDE interface. The main window is titled 'Data Management' and contains a T-SQL console with the command `select * from customers`. Below the console is a grid view of the results, which is a table with columns: CUSTOMERID, CONTACTNAME, COMPANYNAME, CONTACTTITLE, and ADDRESS. The table contains 11 rows of data. On the right side of the window, there is an AI Assistant panel titled 'What can I help you with?'. The panel includes a 'History' section and a text input field for asking questions. The bottom of the window shows a status bar with 'Page 1 of 1' and '1 of 91'.

CUSTOMERID	CONTACTNAME	COMPANYNAME	CONTACTTITLE	ADDRESS
1	ALFKI	Alfreds Futterkiste	Marketing Manager	21 state street
2	ANATR	Ana Trujillo	Ana Trujillo Emparedado	Arda. de la Constitucion
3	ANTON	Antonio Moreno	Antonio Moreno Bateria	Mataderos 2312
4	AROUT	Thomas Hardy	Around the Horn again	Sales Representative 125 Hanover Sq.
5	BERGS	Christina Berglund	Berglunds snabbkop	Order Administrator North Street
6	BLAUS	Hannah Moos	Bill's furniture store	Sales Representative Forsterstr. 57
7	BLONP	Frederique Citeaux	Fred's restaurant	Marketing Manager 24 Kluber place
8	BOLID	Martin Sommer	Bolido Comidas preparadas	Owner C Araquil, 67
9	BONAP	Laurence Lebihan	Bon appetite	Owner 12, rue des Bouchers
10	BOITM	Elizabeth Lincoln	Bottom-Dollar Markets	Accounting Manager 27 tower Road
11	BSBEV	Victoria Ashworth	Fred's cafe	Sales Representative Piccadilly Circus
12	CACTU	Patricio Simpson	Cactus Comidas para llevar	Sales Agentx Cerito 333

## AI powered script editing

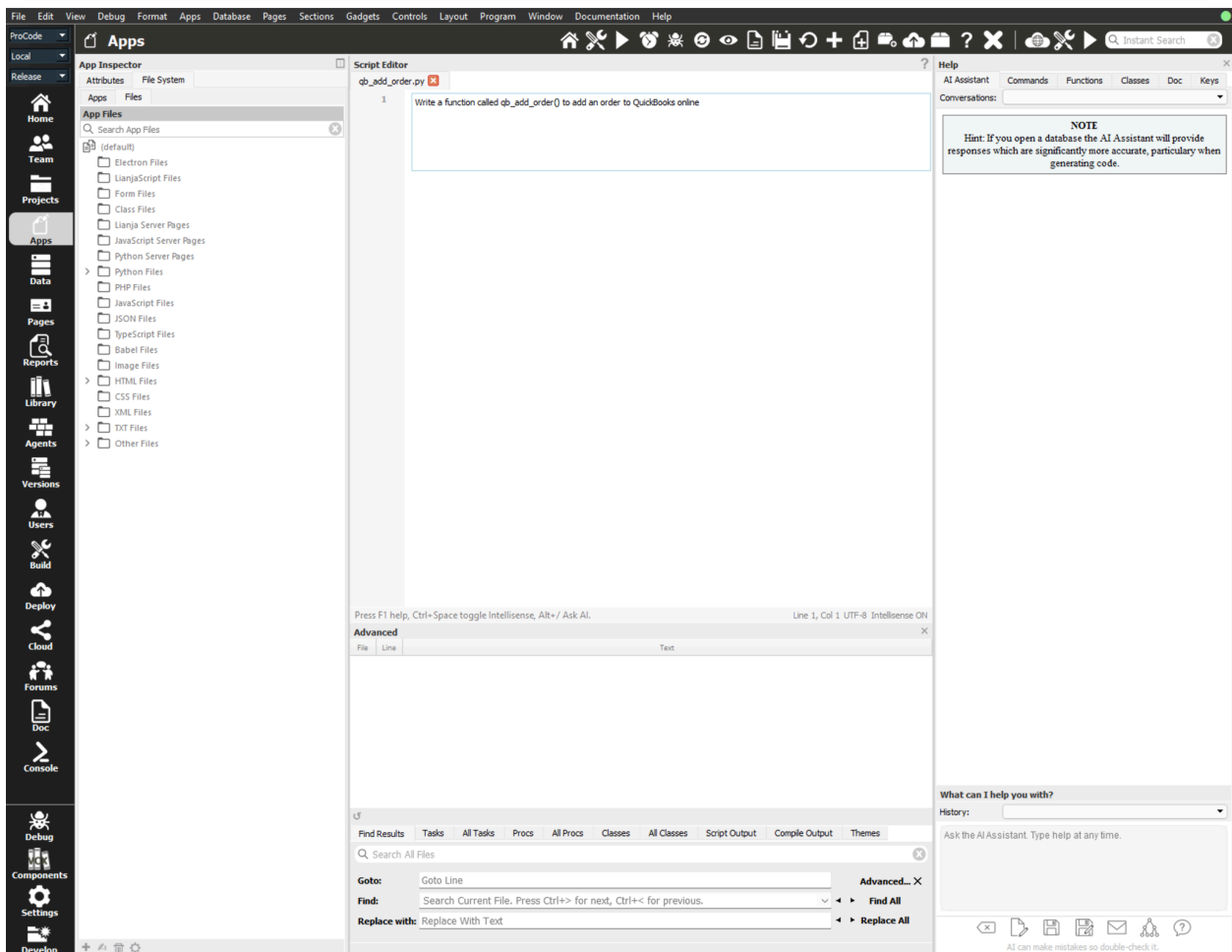
The AI Assistant is an integral part of the script editor in the Apps, Library, Agents and Data workspaces, reducing coding by an order of magnitude. Press Alt+/ and ask the AI Assistant to create the code for you and insert it into the file at the current position.

### Working in the Apps Workspace

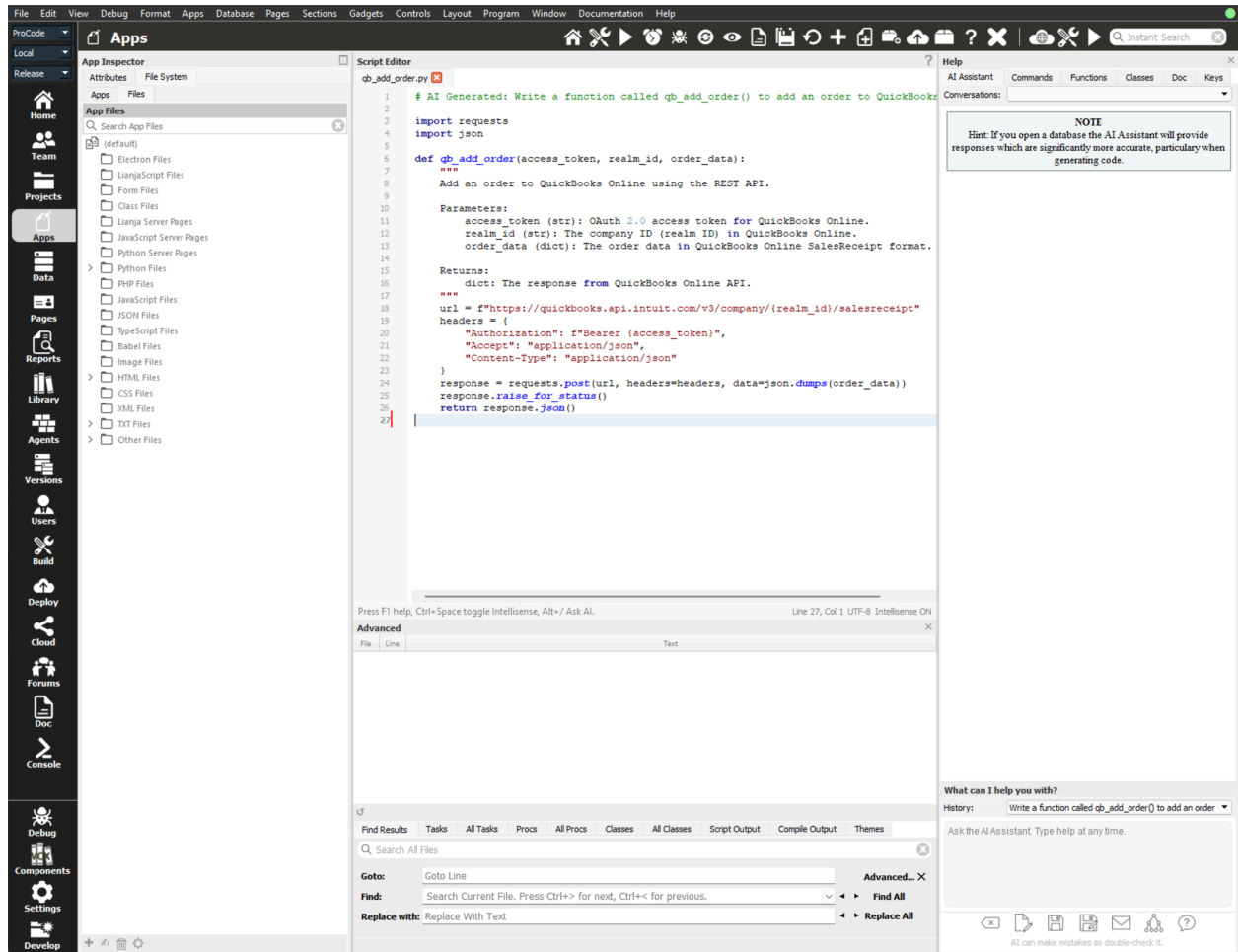
In the Apps or library workplace create a new LianjaScript or Python script called qb\_add\_order. Press Alt+/ then type:

Unset

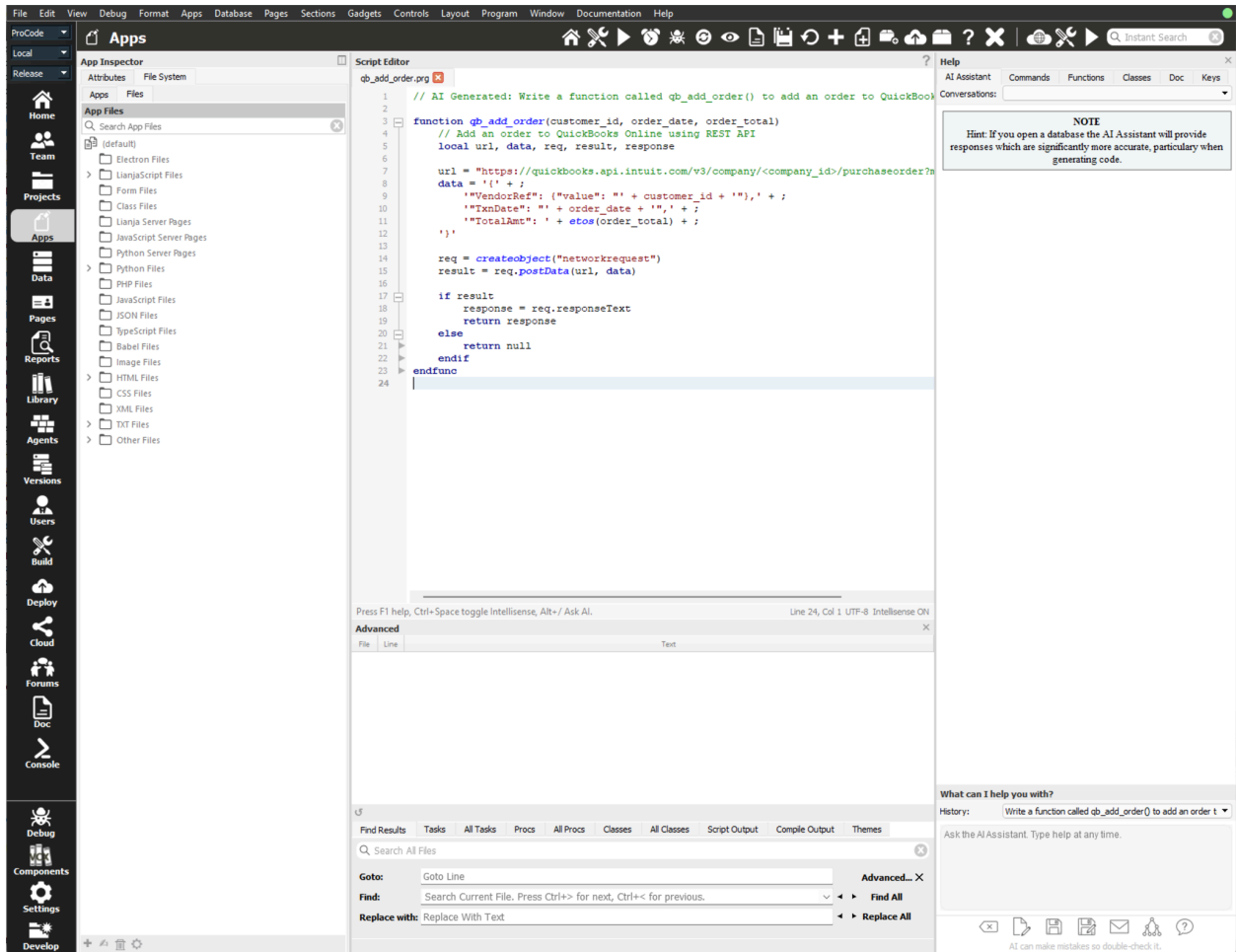
```
Write a function called qb_add_order() to add an order to QuickBooks online
```



Press Return and the Python code will be generated for you.



Similarly in a LianjaScript file.



## Working in the Data Workspace

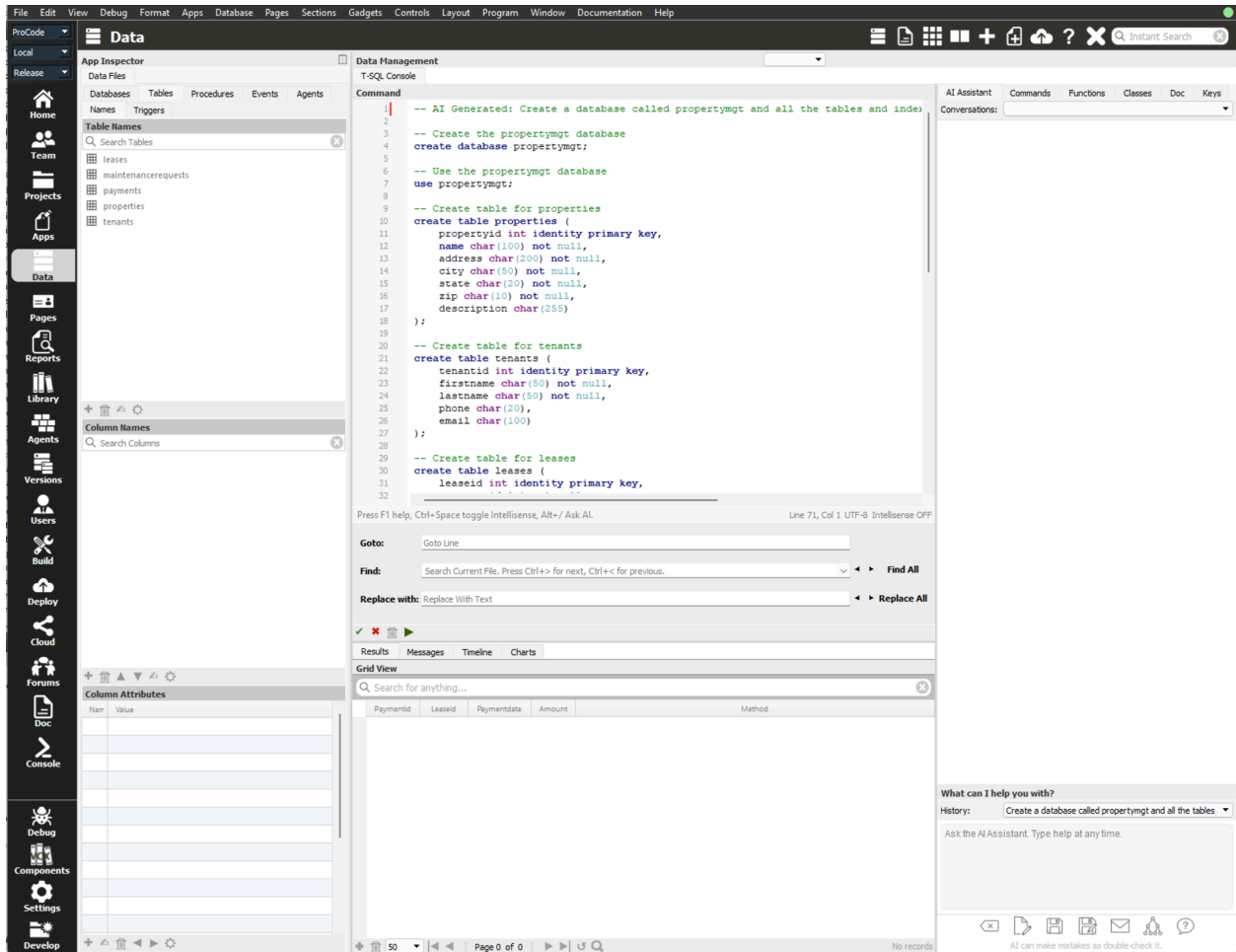
In the T-SQL console.


Press Alt+/ then type:

Unset

Create a database called propertymgt and all the tables and indexes needed to build a property management application





Click the “Execute” icon  and the T-SQL file will be executed.

This is the T-SQL code that was generated then executed to create the database and its tables::

#### SQL

-- AI Generated: Create a database called propertymgt and all the tables and indexes needed to build a property management application

-- Create the database  
create database propertymgt;

-- Use the database  
use propertymgt;

-- Create table for properties  
create table properties (  
 propertyid int identity primary key,  
 name char(100) not null,  
 address char(200) not null,  
 city char(50) not null,  
 state char(20) not null,

```

zip char(10) not null,
description char(255)
);

-- Create index on city for faster searches
create index city on properties(city);

-- Create table for units within properties
create table units (
  unitid int identity primary key,
  propertyid int not null,
  unitnumber char(20) not null,
  bedrooms int not null,
  bathrooms int not null,
  squarefeet int,
  rent decimal(10,2) not null,
  status char(20) not null, -- e.g. Available, Occupied
  constraint fk_units_properties foreign key (propertyid) references properties(propertyid)
);

-- Create index on propertyid for units
create index propertyid on units(propertyid);

-- Create table for tenants
create table tenants (
  tenantid int identity primary key,
  firstname char(50) not null,
  lastname char(50) not null,
  phone char(20),
  email char(100),
  dateofbirth date
);

-- Create index on lastname for tenants
create index lastname on tenants(lastname);

-- Create table for leases
create table leases (
  leaseid int identity primary key,
  unitid int not null,
  tenantid int not null,
  lease_start date not null,
  lease_end date not null,
  rent decimal(10,2) not null,
  securitydeposit decimal(10,2),
  constraint fk_leases_units foreign key (unitid) references units(unitid),
  constraint fk_leases_tenants foreign key (tenantid) references tenants(tenantid)
);

-- Create index on unitid for leases
create index unitid on leases(unitid);

-- Create table for maintenance requests
create table maintenancerequests (
  requestid int identity primary key,
  unitid int not null,
  requestdate date not null,

```

```

description char(255) not null,
status char(20) not null, -- e.g. Open, In Progress, Closed
completiondate date,
constraint fk_maintenancerequests_units foreign key (unitid) references units(unitid)
);

-- Create index on status for maintenance requests
create index status on maintenancerequests(status);

-- Create table for payments
create table payments (
  paymentid int identity primary key,
  leaseid int not null,
  paymentdate date not null,
  amount decimal(10,2) not null,
  paymentmethod char(50), -- e.g. Credit Card, Check, Cash
  constraint fk_payments_leases foreign key (leaseid) references leases(leaseid)
);

-- Create index on leaseid for payments
create index leaseid on payments(leaseid);

```

## AI Assistant deep dive

You chat with the AI Assistant which uses the OpenAI API (Powering ChatGPT) to perform CRUD operations on your *local* private data. This data is under strict security controls based on the logged in users' Roles and Permissions. The AI Assistant is context aware and operates in the following workspaces; Pages, Apps, Library, Data and Agents.

### Pages AI Assistant

The AI Assistant in the Pages workspace can be used to query your data using Natural Language Queries at both development and runtime.

As can be seen from the screenshot below, the AI Assistant is displayed on the right of the window.

Clicking the "Undock" icon on the top right of the AI Assistant will undock it into its own window making more space available for UI design if required.

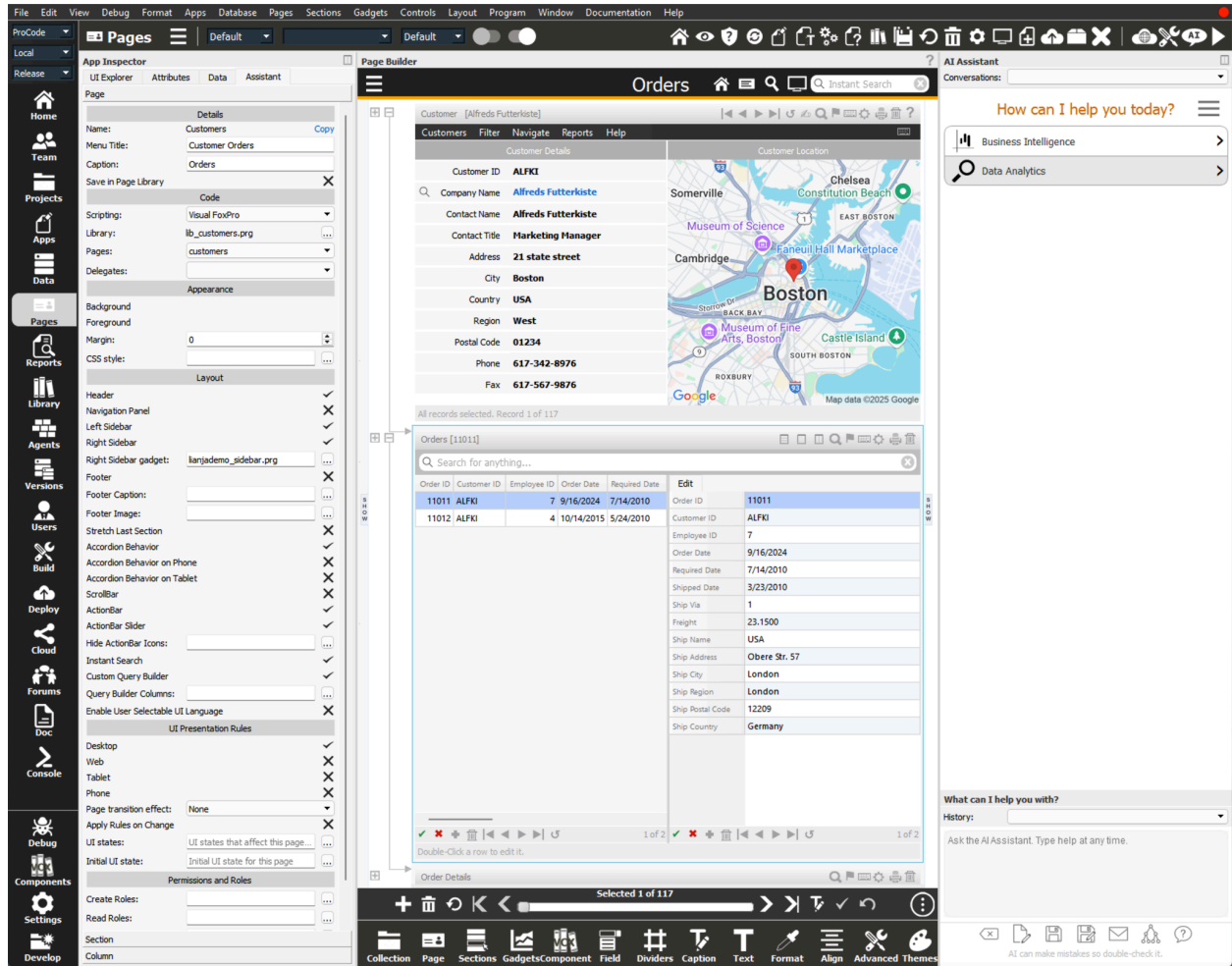
The AI Assistant in the Pages workspace can be toggled on and off by clicking the "AI" icon in the top right of the page.

If you want to disable it completely from either development or runtime mode you can do this in the "App" attributes in the "App Inspector".

## The AI Assistant layout

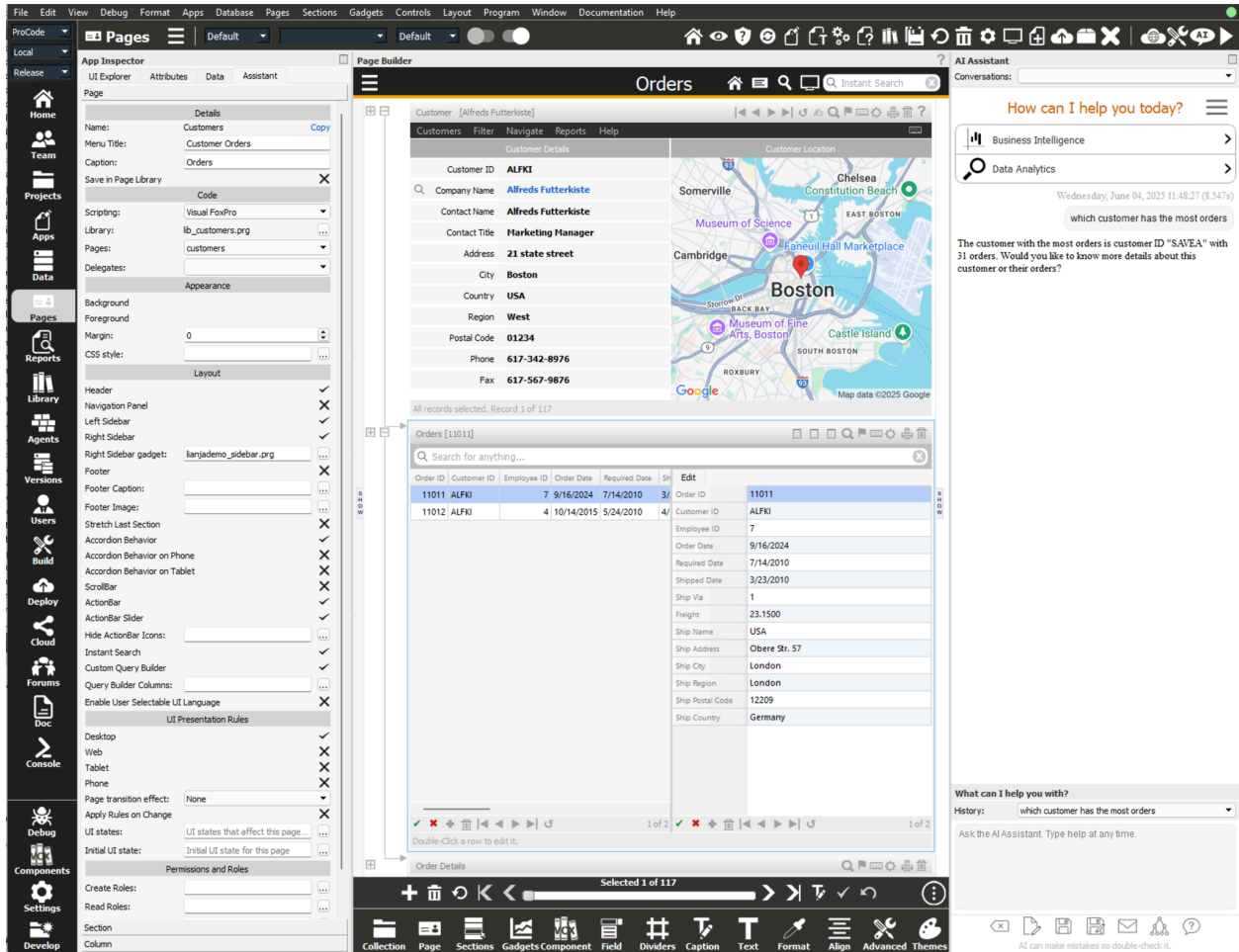
The AI Assistant is laid out from top to bottom like this:

- Header
- Conversations combobox
- Suggestions menu
- Output panel
- Prompt history combobox
- Input panel
- Toolbar
- Progress Bar



## Ask AI anything

With an App open and its database open you can “Ask AI” anything about the data without the need to custom code it in the App.

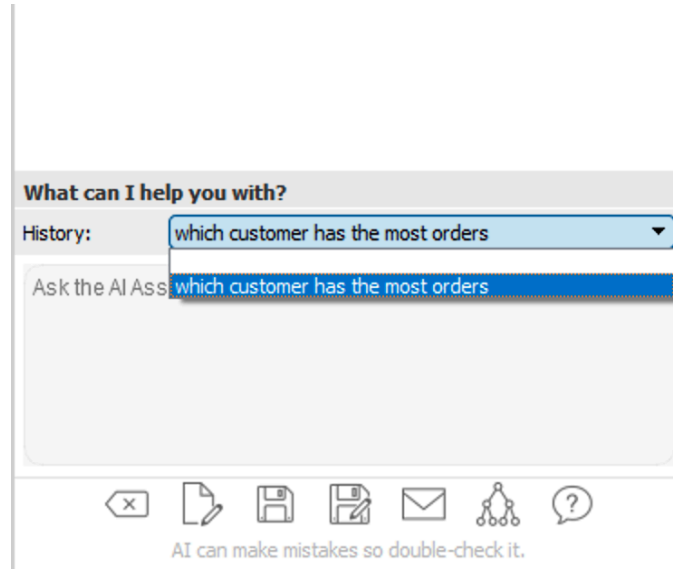


## AI Conversations

As you chat with your data, you can save your current chat history as a “conversation” with a name. This becomes the equivalent of an App. You can replay conversations which replays all of the chat in a conversation. This is a further abstraction on NoCode App building.

## Conversation History

As you ask questions (known as prompts) the history of your questions is maintained in the history combobox. You can select previous prompts and replay them just by selecting them.

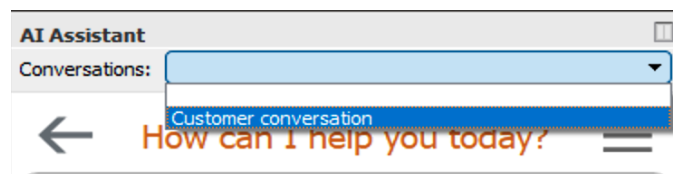


## Saving and Replaying Conversations

To save a conversation click the “Save Conversation” icon  in the AI Assistant toolbar.


Choose a name for the conversation.

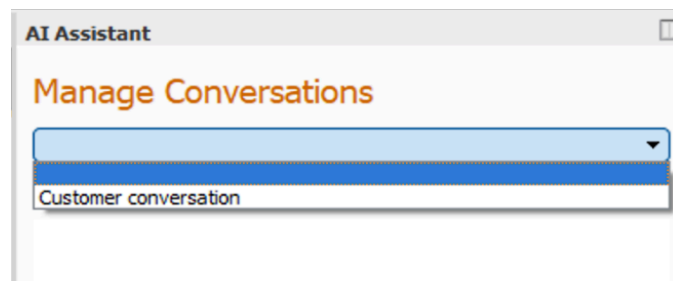
This conversation will persist across Lianja sessions and can be replayed just by selecting it from the Conversations combobox.



## Managing Conversations

You can manage previously saved conversations.

Click the “Manage Conversations” icon  in the AI Assistant toolbar.



Choose a conversation from the combobox at the top and you can edit it, add additional “prompts” or correct prompts.

The text that you want displayed into the output panel of the AI assistant can be different to the prompt that is asked. To accomplish this just postfix the prompt with a # followed by the text you want displayed.

Prompts can include {macros} so that they can be asked in the context of the current “Data View”.

e.g.

Unset

Show me the orders for {customers.customerid}.

## Manage Conversations

Customer conversation

### Conversation

which customer has the most orders  
where does that customer with the most orde  
what is their largest order value  
which customer has the largest order value  
where does that customer with the largest o  
which customer has the least orders

New

Delete

Clear

Done

Cancel

## Email Conversations

At any time during a conversation you can email the conversation to others.

Click the “Email” icon  in the AI Assistant toolbar.

The “Compose Email” panel will overlay on top of the AI Assistant.

You can specify multiple email recipients by separating the email addresses by commas.

If you uncheck the “Heading” checkbox, then the standard heading will not be included in the email.

You can edit the “Message” also.

Click “Send” to send the email to one or more recipients and return to normal AI Assistant operation.

## Compose Email

To:

Cc:

Subject: Conversation History Report

Heading

### Message

Please find attached a Conversation History Report.

Best Regards

admin

### Conversation History

Monday, June 09, 2025 14:05:29 (2.865s)

Average order value

The average order value for the sales database is approximately \$1,635.43. If you need any more details or further analysis, please let me know!

Monday, June 09, 2025 14:05:33 (2.891s)

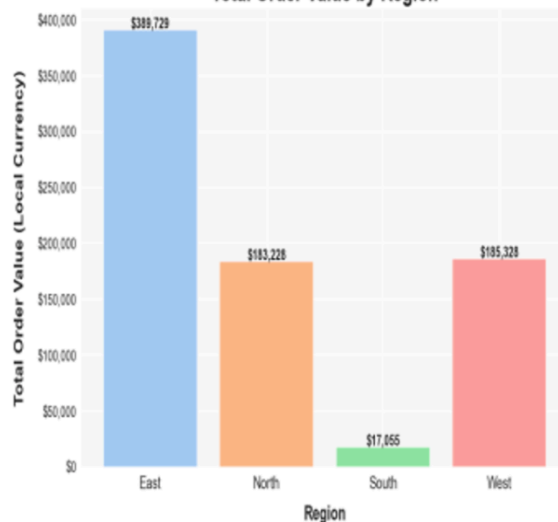
Total number of orders

There are a total of 820 orders in the orders table. If you need any additional details or analysis, please let me know!

Monday, June 09, 2025 14:06:18 (37.396s)

Sales by region


Total Order Value by Region



Cancel

Send

## Understanding Conversation Context

As you ask questions in the AI Assistant the questions and their answers are remembered until you click the  "Clear" icon.

The "Conversations Context" is maintained until you clear it. As you ask questions, they are based on the context of previous questions and their answers.

e.g

Unset

Which customer has the most orders?

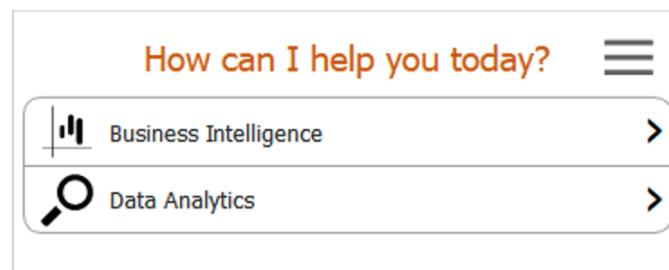
Where do they live?

Without conversation context the second question would make no sense, but when asked in this context the correct address will be displayed.

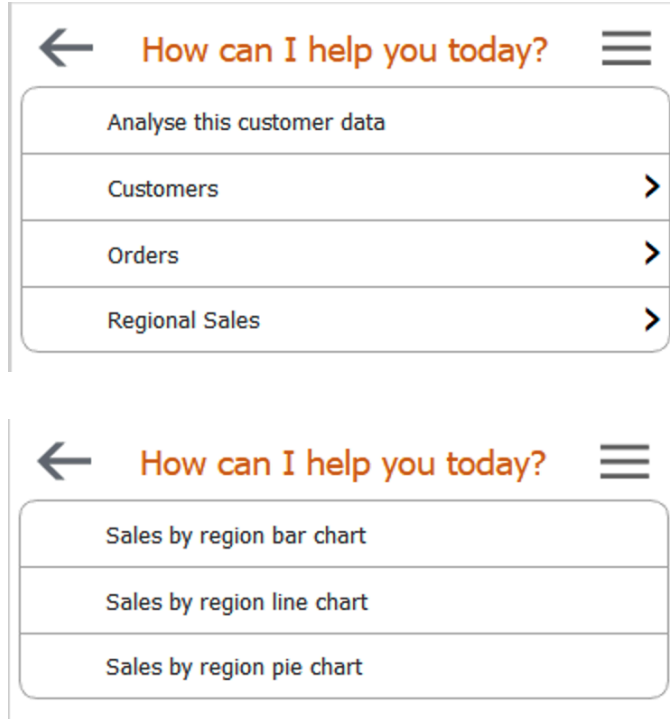
## Suggestion Menus

Suggestion menus provide a hierarchical menu driven interface for users to navigate custom defined questions and actions.

e.g.




Suggestions that have a > on the right have submenus which you can click to navigate down through the menus.

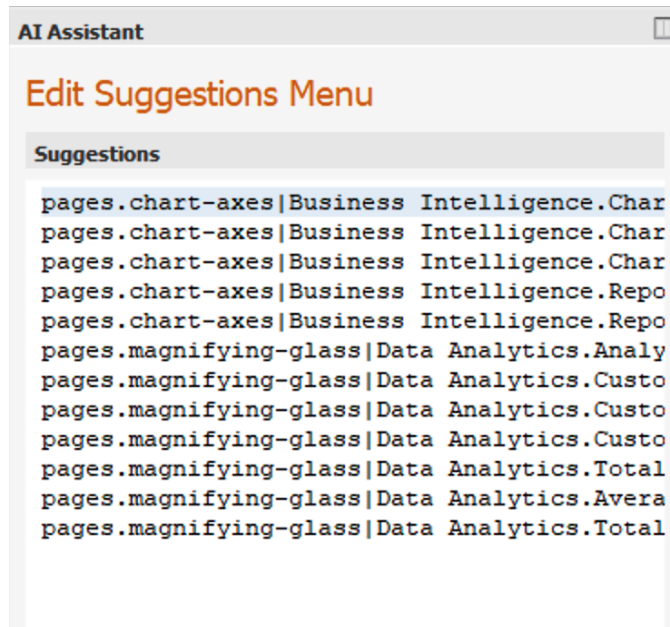


Click the ← “Back” button to navigate backwards to the previous menu.

Click the ≡ “Menu” button to toggle between hiding and showing the suggestions menu.

## Editing Suggestion Menus

You create and/or edit suggestions menus by clicking the  “Edit Suggestions Menu” icon.



The format of the suggestions are as follows:

```
pages.chart-axes|Business Intelligence.Charts.Plot Bar Chart=\
Plot a bar chart for 240,360,120,480|140,340,80,200 title Sample Bar Chart###Plot a bar chart
```

Note that each line in the file can use \ as a line continuation character.

### Optional. only display for the specified database

[database]

**Workspace this suggestion belongs to with an optional icon. Icons come from lib:/icons (pages, data, apps, library or agents)**

```
pages.chart-axes
```

### Dotted path of menu item

```
Business Intelligence.Charts.Plot Bar Chart
```

**The prompt to be sent to the AI Provider. The text following the optional ## is the text to be displayed alongside the AI result in the output panel.**

```
Plot a bar chart for 240,360,120,480|140,340,80,200 title Sample Bar Chart###Plot a bar chart
```

The example suggestions shipped contain:

Unset

```
pages.chart-axes|Business Intelligence.Charts.Plot Bar Chart=\
Plot a bar chart for 240,360,120,480|140,340,80,200 title Sample Bar
Chart###Plot a bar chart
pages.chart-axes|Business Intelligence.Charts.Plot Pie Chart=\
Plot a pie chart for 240,360,120,480 title Sample Pie Chart##Plot a pie chart
pages.chart-axes|Business Intelligence.Charts.Plot Line Chart=\
Plot a line chart for 240,360,120,480|140,340,80,200 title Sample Line
Chart##Plot a line chart
pages.chart-axes|Business Intelligence.Reports.Order Summary=\
show report order summary
pages.chart-axes|Business Intelligence.Reports.Customer Summary=\
show report customers and their orders
pages.magnifying-glass|Data Analytics.Analyse this Dataview=\
Analyse this Dataview
pages.magnifying-glass|Data Analytics.Customers.Customer with most orders=\
which customer has the most orders
pages.magnifying-glass|Data Analytics.Customers.Customer with the largest order
value=\
which customer has the largest order value
pages.magnifying-glass|Data Analytics.Customers.Customer has the least orders=\
which customer has the least orders
pages.magnifying-glass|Data Analytics.Total orders=\
How many orders are there
```

```
pages.magnifying-glass|Data Analytics.Average orders=\
What is the average order value
pages.magnifying-glass|Data Analytics.Total order value=\
What is the total order value
```

## Data AI Assistant

The AI Assistant in the “Data” workspace operates differently than that in the “Pages” workspace.

## AI Powered Business Intelligence

For any prompts (questions) asked in the AI Assistant, it knows about the current database that is open and its tables and columns and will run SQL queries and display the data selected in a grid.

e.g.

```
Unset
```

```
Show me all customers from usa
```

The screenshot shows a data management application with a table of customer information. The table has columns for CUSTOMERID, CONTACTNAME, COMPANYNAME, CONTACTTITLE, and ADDRESS. The AI Assistant chat window on the right shows a conversation where the user asks to show all customers from USA, and the assistant responds that 21 records were selected. Below the chat window, there is a section for Natural Language Queries (NLQ) with an input field containing the text "show me all customers from USA".

CUSTOMERID	CONTACTNAME	COMPANYNAME	CONTACTTITLE	ADDRESS
1	ALFERI	Alfreds Futterkiste	Marketing Manager	21 state street
2	BERGS	Christina Berglund	Berglunds snabbkop	North Street
3	BLAUS	Hannah Moos	Bill's furniture store	Forsterstr. 57
4	BOTTOM	Elizabeth Lincoln	Bottom-Dollar Markets	27 Tower Road
5	FRANK	Peter Franken	Frankensversand	Berliner Platz 43
6	GODOOS	Jose Pedro Freyre	Godos Cocina Tipica	C/ Romero, 33
7	GREAL	Howard Snyder	Great Lakes Food Market	2732 Baker Blvd.
8	HILAA	Carlos Hernandez	HILARION-Abastos	Carrera 22 con Ave. Carlos Soublette 8-35
9	HUNGC	Yoshi Latimer	Hungry Coyote Import Store	City Center Plaza 516 Main St.
10	LAZYK	John Steel	Lazy K Kountry Store	12 Orchestra Terrace
11	LEISS	Jaime Yorres	Let's Stop N Shop	87 Polk St. Suite 5
12	LONEP	Fran Wilson	Lonesome Pine Restaurant	89 Chiaroscuro Rd.
13	MORGK	Alexander Feuer	Morgenstern Gesundkost	Heerstr. 22
14	OLDWO	Rene Phillips	Old World Delicatessen	2743 Bering St.
15	RATIC	Paula Wilson	Rattlesnake Canyon Grocery	Assistant Sales Representative 2817 Milton Dr.
16	SAVEA	Jose Pavarotti	Save-a-lot Markets	187 Suffolk Ln.
17	SPLUR	Art Braunschweiger	Split Rail Beer & Ale	Sales Representative P.O. Box 555
18	THEBI	Liz Nixon	The Big Cheese	Marketing Manager 89 Jefferson Way Suite 2
19	THECR	Liu Wong	The Cracker Box	Marketing Assistant 55 Grizzly Peak Rd.
20	TBAIH	Helviius Nagy	Trail's Head Gourmet Provisioners	Sales Associate 722 Dalvindi Blvd.
21	WHITC	Karl Jablonski	White Clover Markets	Owner 305 - 14th Ave. S. Suite 3B

Once data is displayed in a grid you can drill down into the data using Natural Language Queries.

e.g. in the input panel type:

Unset

Filter containing usa

Or alternatively type in the grid "Search Bar".

Unset

usa

The data in the grid will be filtered and the grid will be re-displayed.

You can continue filtering the data with other filter prompts which are additive to the existing filters.

To reset the filter in the input panel type:

Unset  
Clear filter

The screenshot displays the Data Management interface. The main window shows a table with columns: CUSTOMERID, CONTACTNAME, COMPANYNAME, CONTACTTITLE, and ADDRESS. The table contains 10 rows of data. On the right side, there is an AI Assistant panel with a conversation history. The history shows two prompts: "show me all customers from USA" and "filter containing sales". The AI Assistant also indicates the number of records selected for each prompt. At the bottom of the interface, there is a Messages tab showing the SQL query executed: "select \* from CUSTOMER.L2 SELECT \* FROM customers WHERE country = 'USA' data\_query\_action('action':'filter\_contains','filter\_by':'null','filter\_contains':'sales','order\_by':'null')".

CUSTOMERID	CONTACTNAME	COMPANYNAME	CONTACTTITLE	ADDRESS	
1	BLAUS	Hannah Moos	Bill's furniture store	Sales Representative	Försterstr. 57
2	GODOOS	Jose Pedro Freyre	Godos Cocina Tipica	Sales Manager	C/ Romero, 33
3	HILAA	Carlos Hernandez	HILARION-Abastos	Sales Representative	Carrera 22 con Ave. Carlos Soublette 8-35
4	HUNGC	Yoshi Latimer	Hungry Coyote Import Store	Sales Representative	City Center Plaza 516 Main St.
5	LONEP	Fran Wilson	Lonesome Pine Restaurant	Sales Manager	89 Chiaroscuro Rd.
6	OLDWO	Rene Phillips	Old World Delicatessen	Sales Representative	2743 Bering St.
7	RATC	Paula Wilson	Rattlesnake Canyon Grocery	Assistant Sales Representative	2817 Milton Dr.
8	SAVEA	Jose Pavarotti	Save-a-lot Markets	Sales Representative	187 Suffolk Ln.
9	SPLUR	Art Braunschweiger	Split Rail Beer & Ale	Sales Manager	P.O. Box 555
10	TRAJH	Helvetius Nagy	Trail's Head Gourmet Provisioners	Sales Associate	722 DaVinci Blvd.

Notice how the “Messages” tab at the bottom shows you the SQL being executed.

## Business Intelligence Reporting

You can generate reports in the “Data” workspace which are displayed in a tab alongside your queries and T-SQL stored procedures. You create reports in the “Reports” workspace using the “Report Builder”.

In the AI Assistant input panel type:

Unset

Show report ordersummary

And the report will be displayed.

The screenshot displays a software interface with a sidebar on the left containing navigation icons for Home, Team, Projects, Apps, Data, Reports, Library, Agents, Versions, Users, Build, Deploy, Cloud, Forums, Doc, Console, Debug, Components, Settings, and Develop. The main area is titled 'Data Management' and shows a report for 'Database southwind - Order summary report'. The report includes a table with columns 'Order ID', 'Quantity', 'Unit Price', and 'Discount'. Below the table are two 'Group Summary' sections, each with a table of statistics (Count, Minimum, Maximum, Average, Sub-total) and a donut chart. The donut charts are divided into segments for Red, Blue, Yellow, Green, Purple, and Orange. The AI Assistant panel on the right shows a conversation history with the prompt 'show report ordersummary' and the response 'The report has been generated.' Below the history is an input field with the placeholder text 'What can I help you with?' and a button to 'Ask the AI Assistant. Type help at any time.'

With the tab of the report selected, in the AI Assistant input panel type:

Unset

Filter by orderid 10249

The report will be dynamically filtered and redisplayed.

The screenshot displays a software interface with a sidebar on the left containing navigation icons for Home, Team, Projects, Apps, Data, Reports, Library, Agents, Versions, Users, Build, Deploy, Cloud, Forums, Doc, Console, Debug, and Components. The main area shows a 'Database southwind - Order summary report' with a table of order data, a 'Group Summary (2 rows)' table, and a donut chart. The AI Assistant chat window on the right shows a conversation about generating a report for order ID 10249.

Order ID	Quantity	Unit Price	Discount
10249	10	\$ 18.60	\$ 0.00
10249	40	\$ 42.40	\$ 0.00

Group Summary (2 rows)			
Count	2		
Minimum	0	\$ 0.00	\$ 0.00
Maximum	0	\$ 0.00	\$ 0.00
Average	25	\$ 30.50	\$ 0.00
Sub-total	50	\$ 61.00	\$ 0.00

The donut chart shows the distribution of votes for different categories: Red, Blue, Yellow, Green, Purple, and Orange. The bar chart shows the number of votes for each category.

If you want to see a list of available reports and choose one, type:

Unset  
 Show report

A list of available reports will be displayed in the output panel.

Conversations: Customers from south london

Tuesday, June 10, 2025 09:19:27 (4.583s)

show report

- customers
- employees
- orders
- ordersummary

Choose an option by typing (in this case) a number from 1-4 and the report will be generated and displayed.

## Data Visualization

You can also visualize your data as charts in the “Data” workspace. These are displayed in the “Charts” tab at the bottom.

In the AI Assistant input panel type:

Unset

show me all customers and their orders  
draw me a chart country by count orders

And the chart will be displayed.

The screenshot displays a data management application interface. The main window shows a table of customer orders with columns: CUSTOMERID, COMPANYNAME, CONTACTNAME, COUNTRY, ORDERID, ORDERDATE, SHIPCOUNTRY, and SHIPCITY. The table lists 28 records, including customers like Alfreds Futterkiste, Ana Trujillo Emparedado, Antonio Moreno Taqueria, and Around the Horn again. Below the table, a bar chart titled "Country by Count of OrderID" is displayed, showing the count of orders for various countries. The chart parameters are: Chart: Bar, Title: Country by Count of OrderID, X Column: shipcountry, Y Column: orderid, and Function: cnt. The AI Assistant panel on the right shows the user's input: "show me all customers and their orders" and "draw me a bar chart country by count orders". The assistant's response indicates that 822 records were selected and a bar chart was generated. The interface also includes a sidebar with navigation options like Home, Team, Projects, Apps, Data, Pages, Reports, Library, Agents, Versions, Users, Build, Deploy, Cloud, Forums, Doc, Console, Debug, and Components.

CUSTOMERID	COMPANYNAME	CONTACTNAME	COUNTRY	ORDERID	ORDERDATE	SHIPCOUNTRY	SHIPCITY
1	ALFKI	Alfreds Futterkiste	USA	11011	9/16/2024	Germany	London
2	ALFKI	Alfreds Futterkiste	USA	11012	10/14/2015	Germany	Amsterdam
3	ANATR	Ana Trujillo Emparedado	Mexico	10308	9/20/2011	Mexico	Mexico D.F.
4	ANATR	Ana Trujillo Emparedado	Mexico	10759	11/28/2009	Mexico	Mexico D.F.
5	ANATR	Ana Trujillo Emparedado	Mexico	10926	3/4/2010	Mexico	Mexico D.F.
6	ANTON	Antonio Moreno Taqueria	Mexico	10365	11/27/2011	Mexico	Mexico D.F.
7	ANTON	Antonio Moreno Taqueria	Mexico	10507	10/2/2015	Mexico	Mexico D.F.
8	ANTON	Antonio Moreno Taqueria	Mexico	10535	5/13/2009	Mexico	Mexico D.F.
9	ANTON	Antonio Moreno Taqueria	Mexico	10573	6/19/2009	Mexico	Mexico D.F.
10	ANTON	Antonio Moreno Taqueria	Mexico	10677	9/22/2009	Mexico	Mexico D.F.
11	ANTON	Antonio Moreno Taqueria	Mexico	10856	1/28/2010	Mexico	Mexico D.F.
12	AROUT	Around the Horn again	UK	10355	11/15/2011	UK	Colchester
13	AROUT	Around the Horn again	UK	10383	12/16/2009	UK	Colchester
14	AROUT	Around the Horn again	UK	10453	2/12/2009	UK	Colchester
15	AROUT	Around the Horn again	UK	10558	6/4/2009	UK	Colchester
16	AROUT	Around the Horn again	UK	10707	10/23/2009	UK	Colchester
17	AROUT	Around the Horn again	UK	10741	11/14/2009	UK	Colchester
18	AROUT	Around the Horn again	UK	10743	11/17/2009	UK	Colchester
19	AROUT	Around the Horn again	UK	10768	12/8/2010	UK	Colchester
20	AROUT	Around the Horn again	UK	10793	12/24/2010	UK	Colchester
21	AROUT	Around the Horn again	UK	10864	2/2/2010	UK	Colchester
22	AROUT	Around the Horn again	UK	10920	3/3/2010	UK	Colchester
23	AROUT	Around the Horn again	UK	10953	3/16/2010	UK	Colchester
24	AROUT	Around the Horn again	UK	11016	4/10/2010	UK	Colchester
25	BERGS	Berglunds snabbkop	USA	10278	8/12/2011	Sweden	Lulea
26	BERGS	Berglunds snabbkop	USA	10280	8/14/2011	Sweden	Lulea
27	BERGS	Berglunds snabbkop	USA	10384	12/16/2009	Sweden	Lulea
28	BERGS	Berglunds snabbkop	USA	10444	2/12/2009	Sweden	Lulea

The charts in the “Data” workspace are interactive so you can change the Chart Parameters and click “Apply” to view your data in a variety of ways; Bar, Line, Pie.

You can filter the data in the AI Assistant:

e.g.

```
Unset  
Filter containing "usa,uk,germany"
```

Or alternatively by typing in the grid “Search Bar”:

```
Unset  
usa,uk,germany
```

The data will be filtered and the chart will be re-displayed as the data in the grid is refreshed.

The screenshot displays the 'Data' workspace interface. At the top, there's a menu bar with options like File, Edit, View, Debug, Format, Apps, Database, Pages, Sections, Gadgets, Controls, Layout, Program, Window, Documentation, and Help. Below the menu, the 'Data Management' section shows a T-SQL Console with a query: `usa,uk,germany`. The main area contains a table with columns: CUSTOMERID, COMPANYNAME, CONTACTNAME, COUNTRY, ORDERID, ORDERDATE, SHIPCOUNTRY, and SHIPCITY. The table lists various orders from different companies like 'Alfreds Futterkiste' and 'Berglunds snabbkop' across different countries.

Below the table, a bar chart titled 'Country by Count of OrderID' is displayed. The chart shows the count of orders for each country: Canada (1), France (1), Germany (117), Spain (1), Sweden (37), UK (56), USA (122), and Venezuela (31). The chart parameters are shown on the right, including Chart type (Bar), Title (Country by Count of OrderID), X Column (shipcountry), Y Column (orderid), X Caption (Country), Y Caption (Count of OrderID), and Function (cnt).

On the right side, the AI Assistant chat window is open. It shows a conversation where the user asks to 'show me all customers and their orders' and 'draw a chart country by count orderid'. The AI Assistant responds with the results of the query and the generated chart. The chat history shows the user's previous request: 'draw a chart country by count orderid'.

## What can you do in the Data workspace

At any time in the AI Assistant you can type:

Unset  
help

And a list of built-in shortcut commands will be displayed specific to the specific workspace that you are currently working in. In the case of the “Data” workspace:

Conversations:

help

## Lianja AI Assistant

The Lianja AI Assistant leverages natural language processing (NLP) to process, understand, and generate responses to you in a conversational manner.

The Lianja AI Assistant will help you Build AI Powered Apps with No-Code / Low-Code and provide you with AI powered agentic data analytics and reporting.

Each Lianja workspace i.e. Console, Apps, Data, Pages, Library, Agents have a workspace specific AI Assistant to help you.

You can now build Apps without having to learn the UI of Lianja. Just ask the AI Assistant to guide you.

You chat with the AI Assistant and create *conversations*. At any time you can save conversations *specific to you* and play them back at any time. You can also manage conversations by clicking the icons in the *toolbar* at the bottom of the AI Assistant.

Conversations can be chosen from the combobox at the top of the AI Assistant. Selecting a conversation plays it back for you.

## Data AI Assistant

Command Summary:

- show databases
- show tables
- show reports
- show schema
- show me all *tablename*
- show me all *tablename* for *columnname value*
- join *table1* with *table2*
- filter by *columnname condition*
- filter containing *value*
- remove filter
- orderby *columnname / columnnumber*
- remove orderby
- export as *csv / excel*
- analyse this data
- analyse document *filepath*
- show summary
- hide summary
- show me a [bar/line/pie/doughnut] chart *columnname1* by [sum/average/maximum/minimum] *columnname2*
- show report [*name*]
- show database doc

You can ask the AI Assistant anything.

What can I help you with?

History:

Ask the AI Assistant. Type help at any time.



AI can make mistakes so double-check it.

## Building custom AI Agents

Creation and editing of custom AI Agents can be done with minimum to no coding whatsoever just by asking in the AI Assistant. e.g

Unset

Create an agent in Python to get the current weather?

Unset

Create an agent in Python to send email using sendgrid?

The AI Assistant is multilingual providing support for many languages; English, French, German, Spanish, Portuguese etc. Questions asked will be answered in the same language as they were asked in.

Talk to any Lianja accessible data in real time; native Lianja, MSSQL, MySQL etc! Visualize it in forms, grids, charts, reports and other custom formats. Save it and optionally email the results in a wide variety of formats including excel, csv, xml and custom formats such as invoices!

### Let's build a custom AI Agent

Select the "Agents" workspace.

Custom AI agents can be written in either LianjaScript or Python.

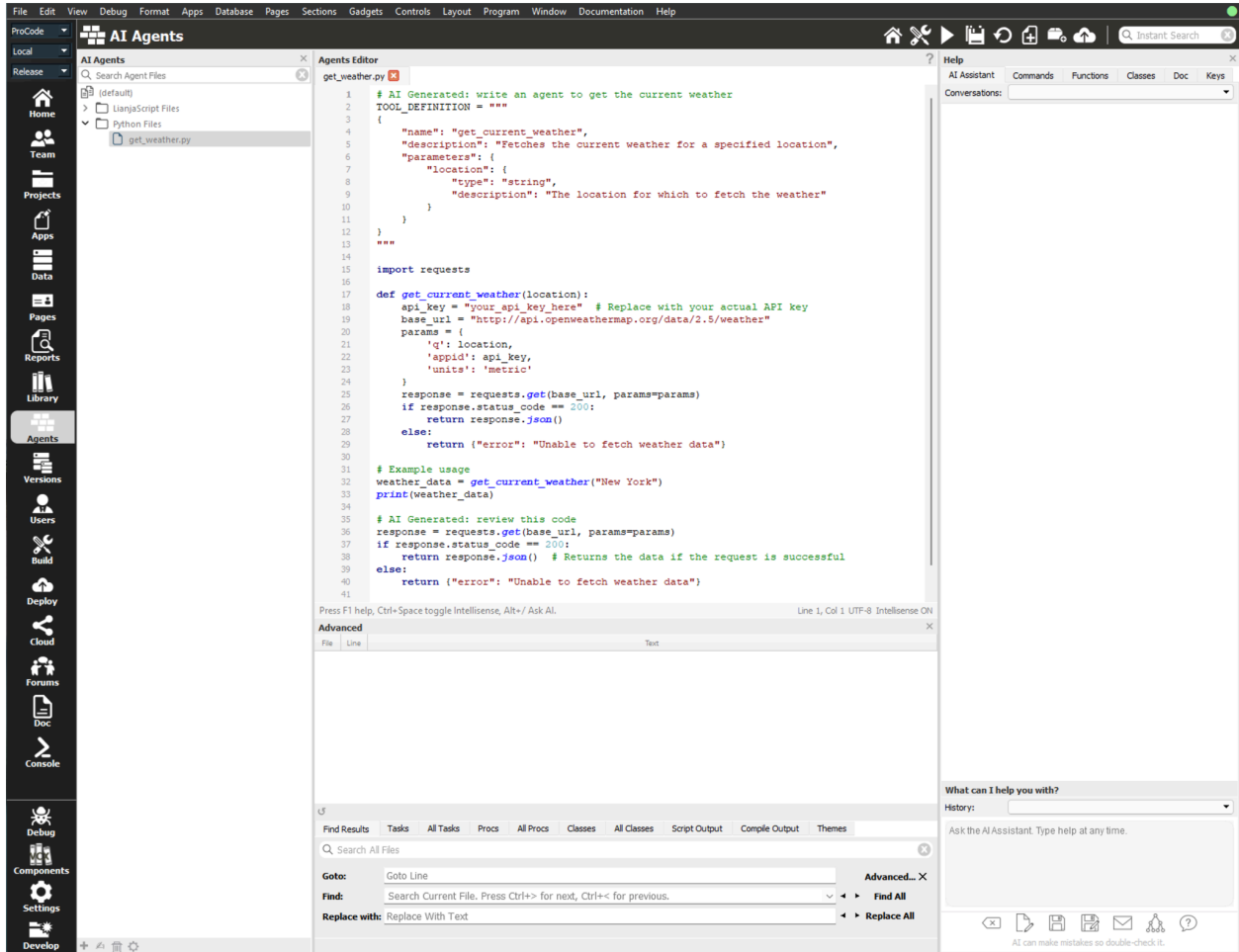
For this example, create a "New" Python file called get\_weather.py.

Press Alt+/ then type:

Unset

Create an agent to get the current weather. Do not add the code for an example call of the agent function?

The Python code will be generated for the agent with the corresponding "TOOL\_DEFINITION" needed by OpenAI.



The Python code generated:

Python

```
# AI Generated: write an agent to get the current weather
TOOL_DEFINITION = """
{
  "name": "get_current_weather",
  "description": "Fetches the current weather for a specified location",
  "parameters": {
    "location": {
      "type": "string",
      "description": "The location for which to fetch the weather"
    }
  }
}
"""

import requests
def get_current_weather(location):
  api_key = "your_api_key_here" # Replace with your actual API key
  base_url = "http://api.openweathermap.org/data/2.5/weather"
  params = {
    'q': location,
  }
  response = requests.get(base_url, params=params)
  if response.status_code == 200:
    return response.json()
  else:
    return {"error": "Unable to fetch weather data"}

# Example usage
weather_data = get_current_weather("New York")
print(weather_data)

# AI Generated: review this code
response = requests.get(base_url, params=params)
if response.status_code == 200:
  return response.json() # Returns the data if the request is successful
else:
  return {"error": "Unable to fetch weather data"}
```

```
'appid': api_key,
'units': 'metric'
}
response = requests.get(base_url, params=params)
if response.status_code == 200:
    return response.json()
else:
    return {"error": "Unable to fetch weather data"}
```

Add the name of the “Agent” to the *agents.conf* file in the “agents” workspace in the format below. Notice how I have prefixed the filename with a ‘-’ sign. This prevents the agent from being used in the AI Assistant *other than* in the “Agents” workspace. After testing you should remove the ‘-’ sign so that the agent is exposed and used by the AI Assistant in all workspaces.

File: agents.conf

Unset

```
-get_weather.py=get_current_weather(location)
```

Format:

**get\_weather.py**

Name of the file containing the agent code

**get\_current\_weather(args...)**

The function in the file to handle the agent call

Note that the name **get\_current\_weather** corresponds to the **name** in the TOOL\_DEFINITION. and **args** correspond to the parameters in the TOOL\_DEFINITION.

## Installing the agent

To test the agent with the AI Assistant you first need to “Install” it.

In the “Agents” workspace HeaderBar, click the “Install Agent” (second from the right).

## Testing the agent

Staying in the “Agents” workspace ask the AI Assistant:

Unset

```
What is the current weather in Boston?
```

If you have an `api_key` in the `get_current_weather()` function your agent will retrieve the current weather in Boston and display it in the AI Assistant output panel.

## Publishing Agents into the Agents repository

Agents can be uploaded into the Agents repo in github. This makes them discoverable by others so that developers can share agents.

Before uploading to github you need to create a manifest file for the agent describing it. This information is used when searching for Agents in the “Agents” workspace.

This manifest file has the same name as the agent file but with a `.json` extension. E.g in this case `get_weather.json`.

This manifest file has the following JSON format.

```
JSON
{
  "name": "get_weather",
  "displayname": "get_weather",
  "description": "Fetches the current weather for a specified location",
  "publisher": "Your name",
  "version": "1.0",
  "tags": "#agent,#weather,#python"
}
```

The “tags” are used to simplify agent search discoverability.

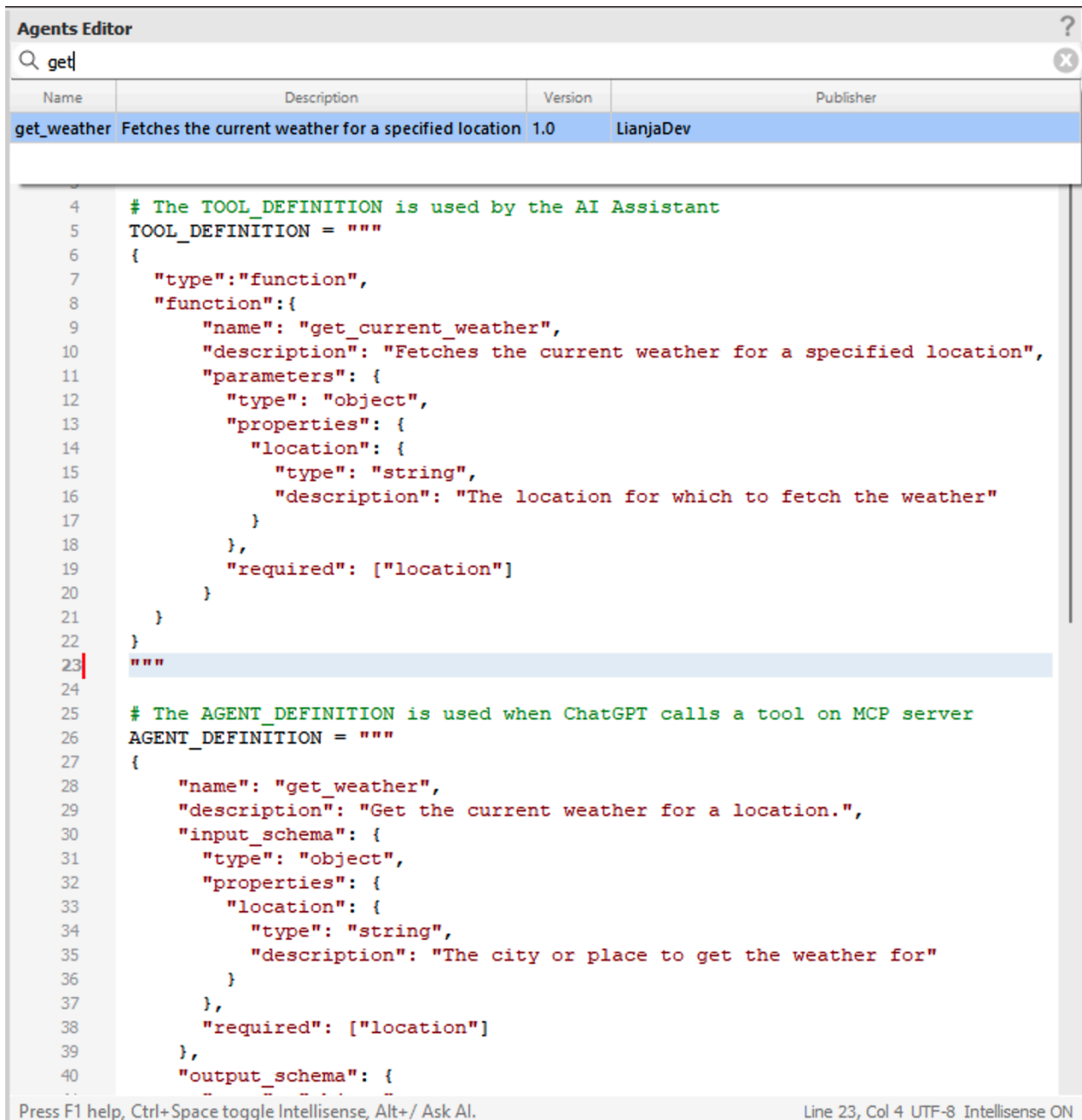
Before uploading you should fill in the Agents Repo Attributes in the App Attributes; including the Personal Access Token assigned to you from the Lianja Dev Team. If you want to publish agents you will need this so contact us at [support@lianja.com](mailto:support@lianja.com). You can of course optionally create your own github account and then create an agents repo private to your company.

## Agent discovery and installation

Agents can be shared by developers by publishing them as described above.

You can search for agents in the Agents repo directly from the editor in the “Agents” workspace. As you type in the agents SearchBar any agents that have #tags or descriptions containing the text you type will be displayed.

Choose an agent by double clicking it in the grid or pressing <return>, and it will be downloaded and installed for you.



The screenshot shows the 'Agents Editor' window. At the top, there is a search bar containing the text 'get'. Below the search bar is a table with the following columns: Name, Description, Version, and Publisher. The table contains one entry: 'get\_weather' with the description 'Fetches the current weather for a specified location', version '1.0', and publisher 'LianjaDev'. Below the table is a code editor showing the JSON definitions for the agent. The code is as follows:

```
4 # The TOOL_DEFINITION is used by the AI Assistant
5 TOOL_DEFINITION = """
6 {
7   "type": "function",
8   "function": {
9     "name": "get_current_weather",
10    "description": "Fetches the current weather for a specified location",
11    "parameters": {
12      "type": "object",
13      "properties": {
14        "location": {
15          "type": "string",
16          "description": "The location for which to fetch the weather"
17        }
18      },
19      "required": ["location"]
20    }
21  }
22 }
23 """
24
25 # The AGENT_DEFINITION is used when ChatGPT calls a tool on MCP server
26 AGENT_DEFINITION = """
27 {
28   "name": "get_weather",
29   "description": "Get the current weather for a location.",
30   "input_schema": {
31     "type": "object",
32     "properties": {
33       "location": {
34         "type": "string",
35         "description": "The city or place to get the weather for"
36       }
37     },
38     "required": ["location"]
39   },
40   "output_schema": {
```

At the bottom of the code editor, there is a status bar that reads: 'Press F1 help, Ctrl+Space toggle Intellisense, Alt+/ Ask AI.' and 'Line 23, Col 4 UTF-8 Intellisense ON'.

After installation of an agent, it will NOT be active on your system until you add it to *agents.conf* as described previously.

## Deploying agents into the Cloud

You deploy agents into the cloud by creating a Lianja package in the “Deploy” workspace of the Lianja App Builder.

After creating a package use the “Lianja Admin Console” on your Lianja Cloud Server instance (or tenancy in the Lianja Cloud) to upload and install the package.

## Limitations and Restrictions

The Lianja 11 App Builder has built-in support for the OpenAI API.

In order to minimize unfair use of the AI capabilities of Lianja 11, there are limits on built-in AI token usage. This is currently set at 100,000 per hour. If you need more than this please contact [sales@lianja.com](mailto:sales@lianja.com).

AI providers such as OpenAI charge for input and output token use.

If you are using the Lianja AI Assistant built-in AI you will be restricted to 100,000 tokens per hour.

There is a progress bar at the very bottom of the AI Assistant which shows you your current usage (hover the mouse over it for details). This resets every hour.

If you want to use more than 100,000 tokens per hour you need to subscribe to the OpenAI API yourself and obtain an API key. This API key should be specified in an environment variable `OPENAI_API_KEY` for all your users, or alternatively specified in the App attributes using the App Inspector.

Note that runtime deployed Apps require you to have your own private OpenAI API key.