

AI Sales Agent



Nikhil Kumar Pandey
Dblytics Project

Real Estate Business

The **real estate business** is a dynamic and multifaceted industry centred around the buying, selling, renting, and management of properties. It includes residential, commercial, industrial, and land-based transactions. People involved in this business—such as real estate agents, brokers, property managers, developers, and investors—juggle a wide variety of tasks on a daily basis.

AI Day-to-Day Tasks in Real Estate Business

1. Client Interaction & Sales

- Handling objections and answering detailed questions.

2. Marketing & Promotion

- Listing properties on websites and MLS.
- Running ad campaigns (online/offline).

3. Property Management (if applicable)

- Collecting rent in online mode.
- Coordinating repairs or maintenance.

4. Market Research & Analysis

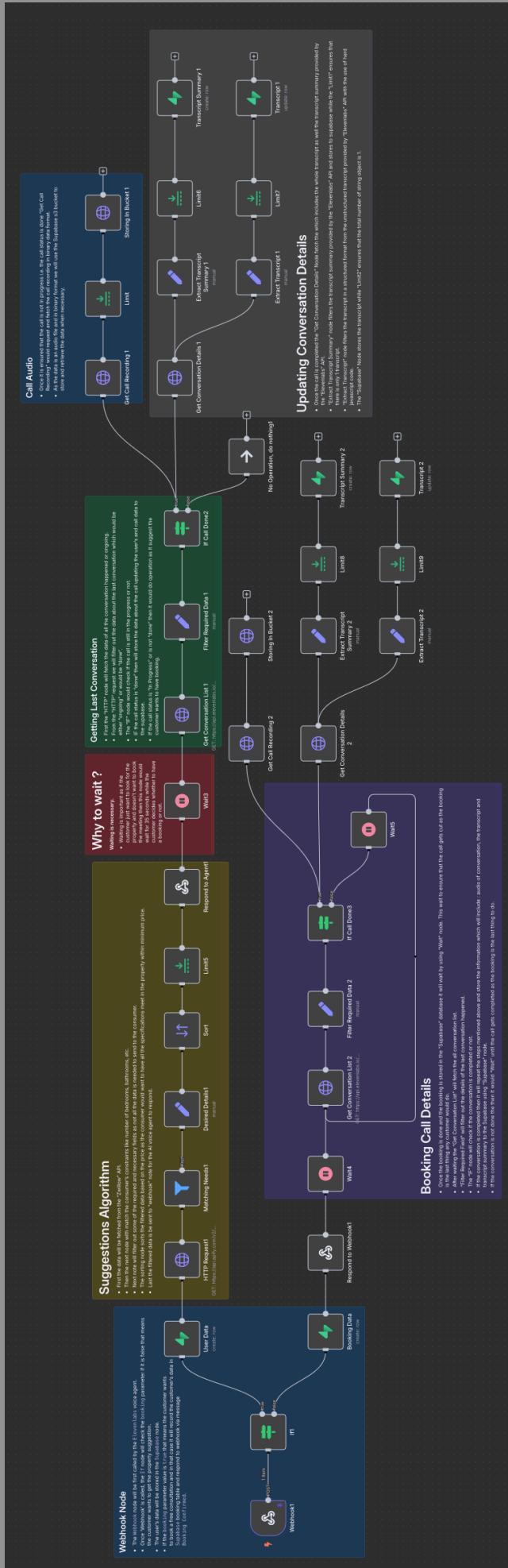
- Analysing property values and price trends.
- Conducting comparative market analysis (CMA).

Real Estate AI Voice Agent helps the Real Estate Business holders to handle clients on everyday basis, running ad campaigns through phones and helps the customer in booking with the real estate agent for the proper one to one consolation.

The Agent helps the customer to get his/her desired property based on the constraints decided by the customer like (number of bedrooms, number of bathrooms, location of the property, budget of the customer and area of the property).

The Agent helps the businesses to run ad campaigns about their company and book one on one consultation with the buyer.

N8N WorkFlow



Elevenlabs

Elevenlabs is an AI tool which provides AI voice models and products powering millions of developers, creators, and enterprises. From low-latency conversational agents to the leading AI voice generator for voiceovers and audiobooks.

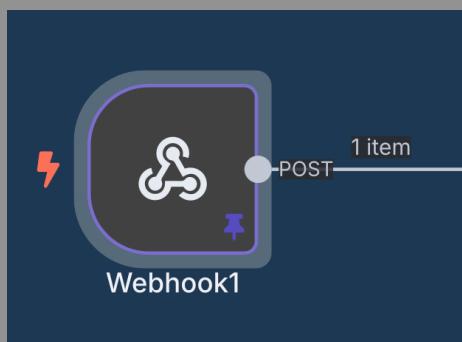
Elevenlabs has been used to connect with the n8n workflow via web-hook functionality in both.

NOTE : Elevenlabs requires a public domain to work, if you not have a public domain it can be easily obtained from ‘ngrok’ with some terms and conditions to run locally on the system.

Elevenlabs Integration : After getting the public domain we create a new “Custom” tool in our created Elevenlabs Conversational AI Agent and use the public domain as the ‘url’ with “POST” method as we want to send the user’s details to the n8n web-hook node. The information we want to send to our “n8n” web-hook node is referenced as body parameters in Elevenlabs, along with the name and description of the parameter. “Save Changes” at the last.

The screenshot shows the 'Edit tool' interface for creating a 'Webhook' tool. The left panel contains basic tool configuration like name, description, method, URL, and response timeout. The right panel is focused on 'Body parameters', where a single property named 'area' is defined with a number type and an LLM prompt value type. A 'Dynamic Variables' section is also visible.

N8N Integration : No authentication is needed, we just need to make the “HTTP” method to “POST” and that’s it. In our workflow the response to this web-hood node is through the “Using Respond to Webhook” node.



The screenshot shows the configuration panel for the 'Webhook1' node. It includes sections for 'Parameters' (with 'Webhook URLs' selected), 'Settings', and 'Docs'. Under 'Parameters', a 'Test URL' is set to https://redirectmeto.com/http://ec2-13-58-27-1 and a 'POST' method is selected. Other fields include 'Path' (sales_agent), 'Authentication' (None), and 'Respond' (Using 'Respond to Webhook' Node). A note says 'Insert a 'Respond to Webhook' node to control when and how you respond. More details'. The 'Options' section shows 'No properties'.

Webhook Node Input : When the web hook is triggered via Elevenlabs voice agent it sends all the collected information about the user by using the tool in which the “url” of n8n web-hook node is POST(ed).

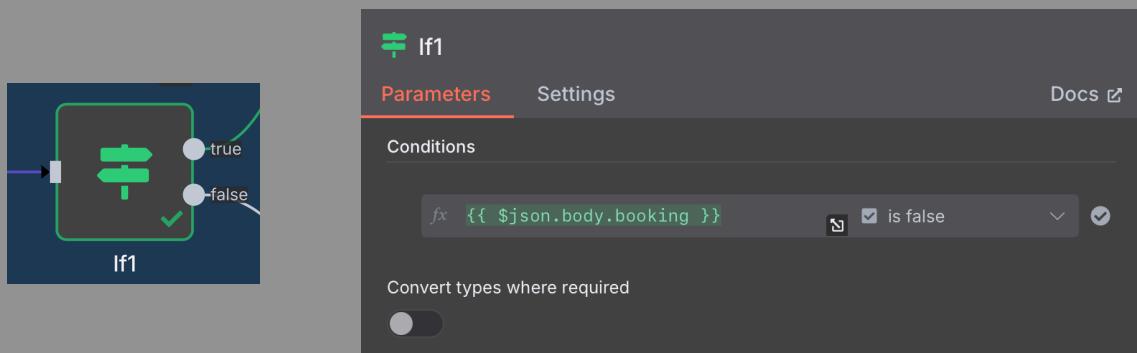
Tool succeeded: n8n_enquiry
0:58
Tool Webhook Result 6.5 s
Webhook call details
Requested URL
POST https://stirring-kingfish-secondly.ngrok-free.app/webhook-test/sales_agent
Request body
Copy {
"name": "Google",
"bathrooms": 1,
"bedrooms": 3,
"price": 2000000,
"email": "google@gmail.com",
"location": "San Francisco",
"booking": false,
"area": 600
}
Parameters extracted by LLM

```
[{"headers": {"host": "stirring-kingfish-secondly.ngrok-free.app", "user-agent": "Python/3.12 aiohttp/3.11.16", "content-length": "121", "accept": "*/*", "accept-encoding": "gzip, deflate", "content-type": "application/json", "x-forwarded-for": "34.67.146.145", "x-forwarded-host": "stirring-kingfish-secondly.ngrok-free.app", "x-forwarded-proto": "https"}, "params": {}, "query": {}, "body": {"price": 750000, "date": "07/06/2025", "email": "deepakjaya@gmail.com", "time": "1:30 PM", "booking": true, "name": "Jaya"}, "webhookUrl": "https://stirring-kingfish-secondly.ngrok-free.app/webhook-test/sales_agent", "executionMode": "test"}]
```

Elevenlabs Webhook

N8N Webhook Call Input

IF Node : It check the “booking” parameter which is one of the body parameter from the elevenlabs voice agent. This parameters tells us whether the customer wants to book a consultation or is just here for the property enquiry. While the value of “booking” parameter is ‘false’ the nodes executed afterwards is for the property suggestion process else nodes executed is for the booking purpose.



Case 1 : Property Suggestion Only

In this case the “booking” parameter always remains ‘false’. While the executable nodes afterwards also have a contingency for the booking that will be discussed later.

Update User Data (Supabase Node : User Data1) :

The users data is updated in the supabase in the respective table.

INPUT
If1
1 item

```
{"accept-encoding": "gzip, deflate", "content-type": "application/json", "x-forwarded-for": "34.67.146.145", "x-forwarded-host": "stirring-kingfish-secondly.ngrok-free.app", "x-forwarded-proto": "https"}, {"params": {}}, {"query": {}}, {"body": {"name": "Jaya", "bathrooms": 1, "bedrooms": 3, "price": 750000, "email": "deepakjaya@gmail.com", "location": "Novato", "booking": false, "area": 570}, "webhookUrl": "https://stirring-kingfish-secondly.ngrok-free.app/webhook-test/sales_agent", "executionMode": "test"}]
```

User Data1
Parameters Settings Docs

Credential to connect with: Restaurent AI Project
Use Custom Schema: Off
Resource: Row
Operation: Create
Table Name or ID: Sales_Customer
Data to Send: Define Below for Each Column
Fields to Send:

Field Name or ID	Field Value
name - (string)	fix {{ \$json.body.name }}
Jaya	
location - (string)	

OUTPUT
1 item

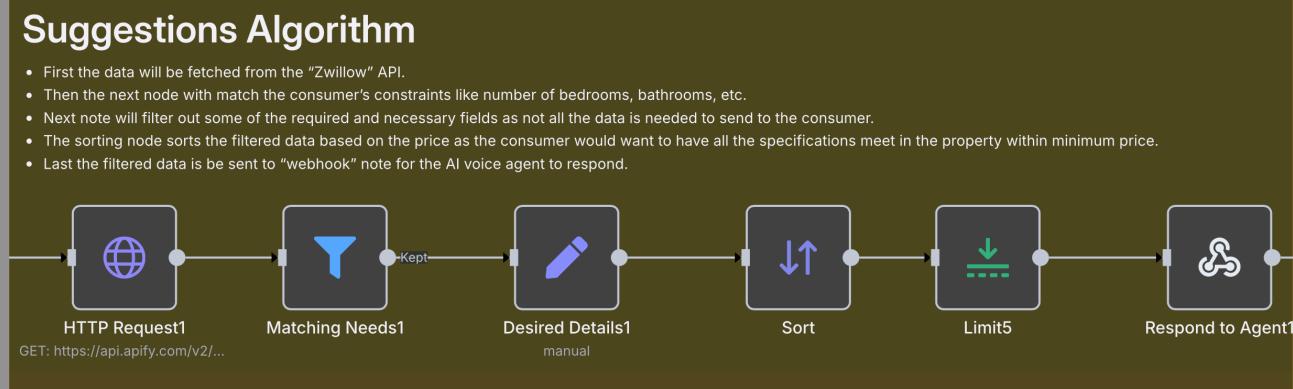
```
[{"name": "Jaya", "location": "Novato", "Email": "deepakjaya@gmail.com", "budget$": 750000, "bedrooms": 3, "bathroom": 1, "area": 570}]
```

I wish this node would...

Update in Supabase table :

	name	Email	location	budget...	bedroo...	bathro...	area
	Jaya	deepakjaya@gmail.com	Novato	750000	3	1	570

Property Suggestions Algorithm :



HTTP Request1 : Fetches the property data from the Zillow api taken from “Apify.com”.

```

fx: {{ $json.hdpData.homeInfo.city }} A is equal to
fx: {{ $('Webhook1').item.json.body.location }}

AND

fx: {{ $json.unformattedPrice }} # is less than or equal to
fx: {{ $('Webhook1').item.json.body.price }}

AND

fx: {{ $json.beds }} # is greater than or equal
fx: {{ $('Webhook1').item.json.body.bedrooms }}

AND

fx: {{ $json.baths }} # is greater than or equal
fx: {{ $('Webhook1').item.json.body.bathrooms }}

```

Matching Needs1 (Filter Node) : It applies the user's constraints with some simple logic like greater than, less than, etc. thus providing us only the details of relevant property.

Field to Set	Value
detailUrl	A String = {{ \$json.detailUrl }} https://www.zillow.com/homedetails/81-Birchwood-Dr-Novato-CA-94947/1...
price	A String = {{ \$json.price }} \$699,000
beds	# Number = {{ \$json.beds }} 3
baths	# Number = {{ \$json.baths }} 2
area	# Number = {{ \$json.area }} 1452
Image Link	A String = {{ \$json.imgSrc }}

Desired Details1 (Edit Field Node) : Not all the data is to be exposed to the user only the necessary details about the property including price, area, address, number of bedrooms, etc are informed to the user.

Type	Simple
Fields To Sort By	Field Name unformattedPrice Enter the field name as text
Order	Ascending

Sort (Sort Node) : Now to give the customer the best result the sorting is used as the customer would want all his demands to be full filled in a minimum amount of money. For this reason the sorting is done based on the price in ascending order with minimum at the top and maximum at the bottom.

Limit (Limit Node) : Only the top 5 results are processed here and responded back to the customer with the help of “**Respond To Webhook**” node, processing all the incoming items into it.

Max Items	5
-----------	---

Verify that the "Webhook" node's "Respond" parameter is set to "Using Respond to Webhook Node". [More details](#)

Respond With	All Incoming Items
Options	No properties

The user is responded with the top property results obtained from running the above algorithm.

Wait Node : This node is for the webhook to again get activated and wait for the POST. This nodes waits for 35 seconds in which the ai agent provides the information about the property and asks the customer whether he/she wants to book a free consultation call. If the call sustains that means the customer wants to either book a call or again he is asking for the system to give property details.

Get Conversation List2 : This is the “HTTP” node with GET request to check whether the call is still in progress or done we first need the conversation ID of the call, for this we fist get all the conversation list from the elevenlabs and filter out the latest conversation ID using the “Edit Field” node.

The image shows two side-by-side node configurations in a workflow editor.

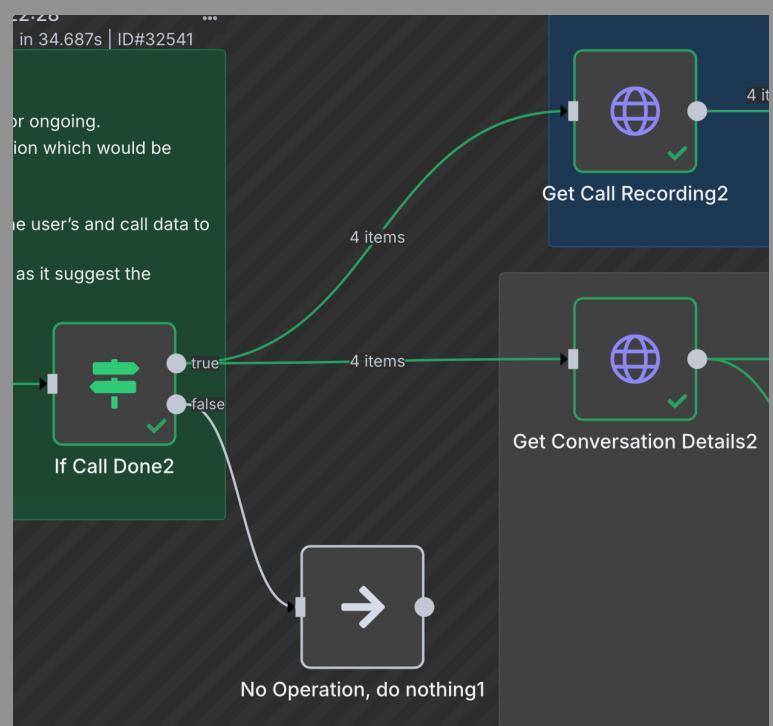
Get Conversation List2 Node Configuration:

- Method:** GET
- URL:** https://api.elevenlabs.io/v1/conval/conversations
- Authentication:** None
- Send Query Parameters:** Off
- Send Headers:** On
- Specify Headers:** Using Fields Below
- Header Parameters:**
 - Name: xi-api-key
 - Value: sk_ccb6430e41226b1ebb9c548ead48f7cb39e00e8c8808a534

Filter Required Data2 Node Configuration:

- Mode:** Manual Mapping
- Fields to Set:**
 - conversation_id:** A String = {{ \$json.conversations[0].conversation_id }} conv_01jx1r1982f80r39en71xj4x78
 - status:** A String = {{ \$json.conversations[0].status }} done
 - duration:** A String = {{ \$json.conversations[0].call_duration_secs }} 34
 - agent_name:** A String = {{ \$json.conversations[0].agent_name }} Sales_Agent

IF : This node checks whether the call is still on progress or the call is done. If the call is still in progress this node goes to do nothing. But if the call is done then it goes to storing call details which includes call recording, transcript and transcript summary.



Call Details Update

Once we insure that the call is not in progress or in the other sense the call is done then call recording is fetched from the Elevenlabs through “Elevenlabs API” via HTTP GET request node. This node requires the conversation id.

After fetching the conversation we get a binary file. This “n8n binary file” is then stored in supabase bucket by using the api and authorisation key from where the call recorded can be retrieved at any time of need.

The screenshot shows a n8n workflow interface with two main nodes:

- Get Call Recording2**: An HTTP GET node. Method: GET. URL: `https://api.elevenlabs.io/v1/convai/conversations/s/{{ $json.conversation_id }}/audio`. Header Parameters: `xi-api-key: sk_ccb6430e41226b1ebb9c548ead48f7cb39e00e8c8808a534`.
- Storing In Bucket2**: A POST node to a Supabase storage bucket. URL: `https://oxfsxmpqjnrewobabj.supabase.co/storage/v1/object/sales-agent-call-recordings/{{ $json.conversation_id }}.mp3`. Header Parameters: `Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpxVCJ9eyJpc3MiOiJzC`.

Between the nodes, there is a data preview window showing the recorded audio file details:

data
File Extension: mp3 Mime Type: audio/mpeg File Size: 539 kB

Get Conversation Details2 (HTTP Node) : This node fetches the conversation details including whole transcript and transcript summary from the elevenlabs using elevenlabs api and conversation id which we retrieved from the HTTP GET request getting conversation list.

The screenshot shows a n8n workflow interface with a single node:

- Get Conversation Details2**: An HTTP GET node. Method: GET. URL: `https://api.elevenlabs.io/v1/convai/conversations/s/{{ $json.conversation_id }}`. Header Parameters: `xi-api-key: sk_ccb6430e41226b1ebb9c548ead48f7cb39e00e8c8808a534`.

To the right, the node's output is displayed in a tree view:

- agent_id**: sqUZLnyVej2jdSWVbd14
- conversation_id**: conv_01jx1r1982f80r39en71x4x78
- status**: done
- transcript**:
 - transcript[0]**
 - transcript[1]**
 - transcript[2]**
 - transcript[3]**:
 - role**: agent
 - message**: [null]
 - tool_calls**:
 - tool_calls[0]**:
 - type**: webhook
 - request_id**: toolu_vrtx_011CUSvxoyQGYWqFJTgZLhW
 - tool_name**: n8n_enquiry
 - params_as_json**: `{"name": "Mark", "email": "mark@gmail.com", "location": "Novato", "bedrooms": 2, "bathrooms": 1, "area": 550, "price": 950000, "booking": false}`
 - tool_has_been_called**: true
 - tool_details**

Supabase Node : These nodes stores the transcript and transcript summary retrieved and processed from the elevenlabs api.

Call Details Update2

Parameters

- Credential to connect with: Restaurant Ai Project
- Use Custom Schema: Off
- Resource: Row
- Operation: Create
- Table Name or ID: Sales_Agent_Call_Details
- Data to Send: Define Below for Each Column
- Fields to Send:

 - Field Name or ID: transcript_summary - (string)
 - Field Value: `fx {{ $json.transcript_summary }}`
 - Eric from Silicon Sales Company offered assistance to Mark, w...

- Field Name or ID: conversation_id - (string)

OUTPUT

1 item

- A email deepakjaya@gmail.com
- A conversation_id conv_01jx1r1982f80r39en71xj4x78
- A transcript_summary Eric from Silicon Sales Company offered assistance to Mark, who is looking for a property in Novato with specific requirements (2 bedrooms, 1 bathroom, 550 sq ft, \$950,000 budget). Eric confirmed Mark's email and attempted to use a tool to find suitable properties, but the tool call failed with a 403 Forbidden error.\n
- A name Jaya
- # id 11
- A transcript [null]

I wish this node would...

Get Conversation Details2

Parameters

- Method: GET
- URL: `https://api.elevenlabs.io/v1/convai/conversations/{{ $json.conversation_id }}`
- Authentication: None
- Send Query Parameters: Off
- Send Headers: On
- Specify Headers: Using Fields Below
- Header Parameters:

 - Name: xi-api-key
 - Value: `fx sk_ccb6430e41226b1ebb9c548ead48f7cb39e00e8c8808a534`

OUTPUT

4 items

- A agent_id sqUZLnyVej2j0SWVbd14
- A conversation_id conv_01jx1r1982f80r39en71xj4x78
- A status done
- transcript

 - transcript[0]
 - transcript[1]
 - transcript[2]
 - transcript[3]

- role agent
- message [null]
- tool_calls

 - tool_calls[0]

- type webhook
- request_id tool_vrtx_011CUSvxoyQGYWqFJTxgZLhW
- tool_name n8n_enquiry
- params_as_json `{"name": "Mark", "email": "mark@gmail.com", "location": "Novato", "bedrooms": 2, "bathrooms": 1, "area": 550, "price": 950000, "booking": false}`
- tool_has_been_called true
- tool_details

Booking Procedure

If the call sustains even after the ‘Wait’ node in the enquiry procedure that means that the customer is now into booking for the free consultation. So now after the “Do Nothing” node our web hook gets activated again waiting to get triggered again.

After the the webhook node gets triggered the “IF” node again checks the value of the “booking” parameter this time the value of this parameter is “True”.

The screenshot shows the configuration of a 'Webhook1' node. On the left, under 'Webhook URLs', a POST method is selected with a test URL pointing to a redirectmeto endpoint. The right side shows the 'OUTPUT' tab with a JSON dump of the webhook payload. The payload includes headers, params, query, and body fields. The 'body' field contains booking details like price, date, email, time, booking status, and name, along with the webhook URL and execution mode.

```
[{"headers": {"host": "stirring-kingfish-secondly.ngrok-free.app", "user-agent": "Python/3.12 aiohttp/3.11.16", "content-length": "121", "accept": "*/*", "accept-encoding": "gzip, deflate", "content-type": "application/json", "x-forwarded-for": "34.67.146.145", "x-forwarded-host": "stirring-kingfish-secondly.ngrok-free.app", "x-forwarded-proto": "https"}, "params": {}}, {"query": {}}, {"body": {"price": 750000, "date": "07/06/2025", "email": "deepakjaya@gmail.com", "time": "1:30 PM", "booking": true, "name": "Jaya"}, "webhookUrl": "https://stirring-kingfish-secondly.ngrok-free.app/webhook-test/sales_agent", "executionMode": "test"}]
```

Booking Data (Supabase Node) : After passing thought the “IF” node the data passes through the “Booking Data” which is supabase node and a new booking gets created in the “Sales_booking” table.

	id	name	email	budget	date	time
4	Steve	Steve	steve@gmail.com	1500000	06/06/2024	1:30 PM
5	Jaya	Jaya	deepakjaya@gmail.com	750000	07/06/2025	1:30 PM

Respond to Webhook : After creating a new row thus completing the booking process the web-hook is responded by the “Respond to webhook1” with a fixed message : “Booking has been completed.”

The screenshot shows the configuration of a 'Respond to Webhook1' node. It includes a 'Parameters' tab with a note about the 'Respond' parameter being set to 'Using Respond to Webhook Node'. Under 'Respond With', there's a 'Text' input field containing the message 'Booking has been completed.'. A note below it says 'When using expressions, note that this node will only run for the first item in the input data'. The 'Response Body' field also contains the same message. The 'Options' tab is empty.

Wait : After creating the booking in the “sales_booking” table this node will wait in order for the call to get completed.

Steps of getting call details will get repeated except for one step and this if this time the call status is “in progress” then it will wait in loop for the call to get finished and once the call gets finished all the steps in getting the call details will get repeated and conversation audio, transcript summary and transcript will get stored in the supabase.

