## **gezi\_noktalari**

| Name | Format | Type | Description |
| --- | --- | --- | --- |
| id | bigint | number |  |
| ad | text | string |  |
| aciklama | text | string |  |
| kategori | text | string |  |
| enlem | double precision | number |  |
| boylam | double precision | number |  |
| fotograf | text | string |  |
| acilis\_saati | time without time zone | string |  |
| kapanis\_saati | time without time zone | string |  |
| telefon\_numarasi | character varying | string |  |
| adres | text | string |  |

## **Read rows**

To read rows in this table, use the select method.

Read all rows

let { data: gezi\_noktalari, error } = await supabase

.from('gezi\_noktalari')

.*select*('\*')

Read specific columns

let { data: gezi\_noktalari, error } = await supabase

.from('gezi\_noktalari')

.*select*('some\_column,other\_column')

Read referenced tables

let { data: gezi\_noktalari, error } = await supabase

.from('gezi\_noktalari')

.*select*(`

some\_column,

other\_table (

foreign\_key

)

`)

With pagination

let { data: gezi\_noktalari, error } = await supabase

.from('gezi\_noktalari')

.*select*('\*')

.*range*(0, 9)

## **Filtering**

Supabase provides a wide range of filters

With filtering

let { data: gezi\_noktalari, error } = await supabase

.from('gezi\_noktalari')

.*select*("\*")

*// Filters*

.*eq*('column', 'Equal to')

.*gt*('column', 'Greater than')

.*lt*('column', 'Less than')

.*gte*('column', 'Greater than or equal to')

.*lte*('column', 'Less than or equal to')

.*like*('column', '%CaseSensitive%')

.*ilike*('column', '%CaseInsensitive%')

.*is*('column', null)

.*in*('column', ['Array', 'Values'])

.*neq*('column', 'Not equal to')

*// Arrays*

.*contains*('array\_column', ['array', 'contains'])

.*containedBy*('array\_column', ['contained', 'by'])

*// Logical operators*

.*not*('column', 'like', 'Negate filter')

.*or*('some\_column.eq.Some value, other\_column.eq.Other value')

## **Insert rows**

insert lets you insert into your tables. You can also insert in bulk and do UPSERT.

insert will also return the replaced values for UPSERT.

Insert a row

const { data, error } = await supabase

.from('gezi\_noktalari')

.*insert*([

{ some\_column: 'someValue', other\_column: 'otherValue' },

])

.*select*()

Insert many rows

const { data, error } = await supabase

.from('gezi\_noktalari')

.*insert*([

{ some\_column: 'someValue' },

{ some\_column: 'otherValue' },

])

.*select*()

Upsert matching rows

const { data, error } = await supabase

.from('gezi\_noktalari')

.*upsert*({ some\_column: 'someValue' })

.*select*()

## **Update rows**

update lets you update rows. update will match all rows by default. You can update specific rows using horizontal filters, e.g. eq, lt, and is.

update will also return the replaced values for UPDATE.

Update matching rows

const { data, error } = await supabase

.from('gezi\_noktalari')

.*update*({ other\_column: 'otherValue' })

.*eq*('some\_column', 'someValue')

.*select*()

## **Delete rows**

delete lets you delete rows. delete will match all rows by default, so remember to specify your filters!

Delete matching rows

const { error } = await supabase

.from('gezi\_noktalari')

.*delete*()

.*eq*('some\_column', 'someValue')

## **Subscribe to changes**

Supabase provides realtime functionality and broadcasts database changes to authorized users depending on Row Level Security (RLS) policies.

Subscribe to all events

const channels = supabase.*channel*('custom-all-channel')

.*on*(

'postgres\_changes',

{ event: '\*', schema: 'public', table: 'gezi\_noktalari' },

(payload) => {

console.*log*('Change received!', payload)

}

)

.*subscribe*()

Subscribe to inserts

const channels = supabase.*channel*('custom-insert-channel')

.*on*(

'postgres\_changes',

{ event: 'INSERT', schema: 'public', table: 'gezi\_noktalari' },

(payload) => {

console.*log*('Change received!', payload)

}

)

.*subscribe*()

Subscribe to updates

const channels = supabase.*channel*('custom-update-channel')

.*on*(

'postgres\_changes',

{ event: 'UPDATE', schema: 'public', table: 'gezi\_noktalari' },

(payload) => {

console.*log*('Change received!', payload)

}

)

.*subscribe*()

Subscribe to deletes

const channels = supabase.*channel*('custom-delete-channel')

.*on*(

'postgres\_changes',

{ event: 'DELETE', schema: 'public', table: 'gezi\_noktalari' },

(payload) => {

console.*log*('Change received!', payload)

}

)

.*subscribe*()

Subscribe to specific rows

const channels = supabase.*channel*('custom-filter-channel')

.*on*(

'postgres\_changes',

{ event: '\*', schema: 'public', table: 'gezi\_noktalari', filter: 'some\_column=eq.some\_value' },

(payload) => {

console.*log*('Change received!', payload)

}

)

.*subscribe*()

## **rotalar**

| Name | Format | Type | Description |
| --- | --- | --- | --- |
| id | bigint | number |  |
| aciklama | text | string |  |
| gezi\_nok\_id | bigint | number |  |
| kategori | text | string |  |

## **Read rows**

To read rows in this table, use the select method.

Read all rows

let { data: rotalar, error } = await supabase

.from('rotalar')

.*select*('\*')

Read specific columns

let { data: rotalar, error } = await supabase

.from('rotalar')

.*select*('some\_column,other\_column')

Read referenced tables

let { data: rotalar, error } = await supabase

.from('rotalar')

.*select*(`

some\_column,

other\_table (

foreign\_key

)

`)

With pagination

let { data: rotalar, error } = await supabase

.from('rotalar')

.*select*('\*')

.*range*(0, 9)

## **Filtering**

Supabase provides a wide range of filters

With filtering

let { data: rotalar, error } = await supabase

.from('rotalar')

.*select*("\*")

*// Filters*

.*eq*('column', 'Equal to')

.*gt*('column', 'Greater than')

.*lt*('column', 'Less than')

.*gte*('column', 'Greater than or equal to')

.*lte*('column', 'Less than or equal to')

.*like*('column', '%CaseSensitive%')

.*ilike*('column', '%CaseInsensitive%')

.*is*('column', null)

.*in*('column', ['Array', 'Values'])

.*neq*('column', 'Not equal to')

*// Arrays*

.*contains*('array\_column', ['array', 'contains'])

.*containedBy*('array\_column', ['contained', 'by'])

*// Logical operators*

.*not*('column', 'like', 'Negate filter')

.*or*('some\_column.eq.Some value, other\_column.eq.Other value')

## **Insert rows**

insert lets you insert into your tables. You can also insert in bulk and do UPSERT.

insert will also return the replaced values for UPSERT.

Insert a row

const { data, error } = await supabase

.from('rotalar')

.*insert*([

{ some\_column: 'someValue', other\_column: 'otherValue' },

])

.*select*()

Insert many rows

const { data, error } = await supabase

.from('rotalar')

.*insert*([

{ some\_column: 'someValue' },

{ some\_column: 'otherValue' },

])

.*select*()

Upsert matching rows

const { data, error } = await supabase

.from('rotalar')

.*upsert*({ some\_column: 'someValue' })

.*select*()

## **Update rows**

update lets you update rows. update will match all rows by default. You can update specific rows using horizontal filters, e.g. eq, lt, and is.

update will also return the replaced values for UPDATE.

Update matching rows

const { data, error } = await supabase

.from('rotalar')

.*update*({ other\_column: 'otherValue' })

.*eq*('some\_column', 'someValue')

.*select*()

## **Delete rows**

delete lets you delete rows. delete will match all rows by default, so remember to specify your filters!

Delete matching rows

const { error } = await supabase

.from('rotalar')

.*delete*()

.*eq*('some\_column', 'someValue')

## **Subscribe to changes**

Supabase provides realtime functionality and broadcasts database changes to authorized users depending on Row Level Security (RLS) policies.

Subscribe to all events

const channels = supabase.*channel*('custom-all-channel')

.*on*(

'postgres\_changes',

{ event: '\*', schema: 'public', table: 'rotalar' },

(payload) => {

console.*log*('Change received!', payload)

}

)

.*subscribe*()

Subscribe to inserts

const channels = supabase.*channel*('custom-insert-channel')

.*on*(

'postgres\_changes',

{ event: 'INSERT', schema: 'public', table: 'rotalar' },

(payload) => {

console.*log*('Change received!', payload)

}

)

.*subscribe*()

Subscribe to updates

const channels = supabase.*channel*('custom-update-channel')

.*on*(

'postgres\_changes',

{ event: 'UPDATE', schema: 'public', table: 'rotalar' },

(payload) => {

console.*log*('Change received!', payload)

}

)

.*subscribe*()

Subscribe to deletes

const channels = supabase.*channel*('custom-delete-channel')

.*on*(

'postgres\_changes',

{ event: 'DELETE', schema: 'public', table: 'rotalar' },

(payload) => {

console.*log*('Change received!', payload)

}

)

.*subscribe*()

Subscribe to specific rows

const channels = supabase.*channel*('custom-filter-channel')

.*on*(

'postgres\_changes',

{ event: '\*', schema: 'public', table: 'rotalar', filter: 'some\_column=eq.some\_value' },

(payload) => {

console.*log*('Change received!', payload)

}

)

.*subscribe*()

## **influencerlar**

| Name | Format | Type | Description |
| --- | --- | --- | --- |
| id | bigint | number |  |
| ad | text | string |  |
| aciklama | text | string |  |
| fotograf | text | string |  |
| instagram\_linki | text | string |  |
| rota\_id | bigint | number |  |

## **Read rows**

To read rows in this table, use the select method.

Read all rows

let { data: influencerlar, error } = await supabase

.from('influencerlar')

.*select*('\*')

Read specific columns

let { data: influencerlar, error } = await supabase

.from('influencerlar')

.*select*('some\_column,other\_column')

Read referenced tables

let { data: influencerlar, error } = await supabase

.from('influencerlar')

.*select*(`

some\_column,

other\_table (

foreign\_key

)

`)

With pagination

let { data: influencerlar, error } = await supabase

.from('influencerlar')

.*select*('\*')

.*range*(0, 9)

## **Filtering**

Supabase provides a wide range of filters

With filtering

let { data: influencerlar, error } = await supabase

.from('influencerlar')

.*select*("\*")

*// Filters*

.*eq*('column', 'Equal to')

.*gt*('column', 'Greater than')

.*lt*('column', 'Less than')

.*gte*('column', 'Greater than or equal to')

.*lte*('column', 'Less than or equal to')

.*like*('column', '%CaseSensitive%')

.*ilike*('column', '%CaseInsensitive%')

.*is*('column', null)

.*in*('column', ['Array', 'Values'])

.*neq*('column', 'Not equal to')

*// Arrays*

.*contains*('array\_column', ['array', 'contains'])

.*containedBy*('array\_column', ['contained', 'by'])

*// Logical operators*

.*not*('column', 'like', 'Negate filter')

.*or*('some\_column.eq.Some value, other\_column.eq.Other value')

## **Insert rows**

insert lets you insert into your tables. You can also insert in bulk and do UPSERT.

insert will also return the replaced values for UPSERT.

Insert a row

const { data, error } = await supabase

.from('influencerlar')

.*insert*([

{ some\_column: 'someValue', other\_column: 'otherValue' },

])

.*select*()

Insert many rows

const { data, error } = await supabase

.from('influencerlar')

.*insert*([

{ some\_column: 'someValue' },

{ some\_column: 'otherValue' },

])

.*select*()

Upsert matching rows

const { data, error } = await supabase

.from('influencerlar')

.*upsert*({ some\_column: 'someValue' })

.*select*()

## **Update rows**

update lets you update rows. update will match all rows by default. You can update specific rows using horizontal filters, e.g. eq, lt, and is.

update will also return the replaced values for UPDATE.

Update matching rows

const { data, error } = await supabase

.from('influencerlar')

.*update*({ other\_column: 'otherValue' })

.*eq*('some\_column', 'someValue')

.*select*()

## **Delete rows**

delete lets you delete rows. delete will match all rows by default, so remember to specify your filters!

Delete matching rows

const { error } = await supabase

.from('influencerlar')

.*delete*()

.*eq*('some\_column', 'someValue')

## **Subscribe to changes**

Supabase provides realtime functionality and broadcasts database changes to authorized users depending on Row Level Security (RLS) policies.

Subscribe to all events

const channels = supabase.*channel*('custom-all-channel')

.*on*(

'postgres\_changes',

{ event: '\*', schema: 'public', table: 'influencerlar' },

(payload) => {

console.*log*('Change received!', payload)

}

)

.*subscribe*()

Subscribe to inserts

const channels = supabase.*channel*('custom-insert-channel')

.*on*(

'postgres\_changes',

{ event: 'INSERT', schema: 'public', table: 'influencerlar' },

(payload) => {

console.*log*('Change received!', payload)

}

)

.*subscribe*()

Subscribe to updates

const channels = supabase.*channel*('custom-update-channel')

.*on*(

'postgres\_changes',

{ event: 'UPDATE', schema: 'public', table: 'influencerlar' },

(payload) => {

console.*log*('Change received!', payload)

}

)

.*subscribe*()

Subscribe to deletes

const channels = supabase.*channel*('custom-delete-channel')

.*on*(

'postgres\_changes',

{ event: 'DELETE', schema: 'public', table: 'influencerlar' },

(payload) => {

console.*log*('Change received!', payload)

}

)

.*subscribe*()

Subscribe to specific rows

const channels = supabase.*channel*('custom-filter-channel')

.*on*(

'postgres\_changes',

{ event: '\*', schema: 'public', table: 'influencerlar', filter: 'some\_column=eq.some\_value' },

(payload) => {

console.*log*('Change received!', payload)

}

)

.*subscribe*()

**etkinlikler**

| Name | Format | Type | Description |
| --- | --- | --- | --- |
| id | bigint | number |  |
| ad | text | string |  |
| aciklama | text | string |  |
| fotograf | text | string |  |
| tarih | date | string |  |
| enlem | double precision | number |  |
| boylam | double precision | number |  |
| telefon\_numarasi | character varying | string |  |
| bilet\_linki | text | string |  |
| adres | text | string |  |

## **Read rows**

To read rows in this table, use the select method.

Read all rows

let { data: etkinlikler, error } = await supabase

.from('etkinlikler')

.*select*('\*')

Read specific columns

let { data: etkinlikler, error } = await supabase

.from('etkinlikler')

.*select*('some\_column,other\_column')

Read referenced tables

let { data: etkinlikler, error } = await supabase

.from('etkinlikler')

.*select*(`

some\_column,

other\_table (

foreign\_key

)

`)

With pagination

let { data: etkinlikler, error } = await supabase

.from('etkinlikler')

.*select*('\*')

.*range*(0, 9)

## **Filtering**

Supabase provides a wide range of filters

With filtering

let { data: etkinlikler, error } = await supabase

.from('etkinlikler')

.*select*("\*")

*// Filters*

.*eq*('column', 'Equal to')

.*gt*('column', 'Greater than')

.*lt*('column', 'Less than')

.*gte*('column', 'Greater than or equal to')

.*lte*('column', 'Less than or equal to')

.*like*('column', '%CaseSensitive%')

.*ilike*('column', '%CaseInsensitive%')

.*is*('column', null)

.*in*('column', ['Array', 'Values'])

.*neq*('column', 'Not equal to')

*// Arrays*

.*contains*('array\_column', ['array', 'contains'])

.*containedBy*('array\_column', ['contained', 'by'])

*// Logical operators*

.*not*('column', 'like', 'Negate filter')

.*or*('some\_column.eq.Some value, other\_column.eq.Other value')

## **Insert rows**

insert lets you insert into your tables. You can also insert in bulk and do UPSERT.

insert will also return the replaced values for UPSERT.

Insert a row

const { data, error } = await supabase

.from('etkinlikler')

.*insert*([

{ some\_column: 'someValue', other\_column: 'otherValue' },

])

.*select*()

Insert many rows

const { data, error } = await supabase

.from('etkinlikler')

.*insert*([

{ some\_column: 'someValue' },

{ some\_column: 'otherValue' },

])

.*select*()

Upsert matching rows

const { data, error } = await supabase

.from('etkinlikler')

.*upsert*({ some\_column: 'someValue' })

.*select*()

## **Update rows**

update lets you update rows. update will match all rows by default. You can update specific rows using horizontal filters, e.g. eq, lt, and is.

update will also return the replaced values for UPDATE.

Update matching rows

const { data, error } = await supabase

.from('etkinlikler')

.*update*({ other\_column: 'otherValue' })

.*eq*('some\_column', 'someValue')

.*select*()

## **Delete rows**

delete lets you delete rows. delete will match all rows by default, so remember to specify your filters!

Delete matching rows

const { error } = await supabase

.from('etkinlikler')

.*delete*()

.*eq*('some\_column', 'someValue')

## **Subscribe to changes**

Supabase provides realtime functionality and broadcasts database changes to authorized users depending on Row Level Security (RLS) policies.

Subscribe to all events

const channels = supabase.*channel*('custom-all-channel')

.*on*(

'postgres\_changes',

{ event: '\*', schema: 'public', table: 'etkinlikler' },

(payload) => {

console.*log*('Change received!', payload)

}

)

.*subscribe*()

Subscribe to inserts

const channels = supabase.*channel*('custom-insert-channel')

.*on*(

'postgres\_changes',

{ event: 'INSERT', schema: 'public', table: 'etkinlikler' },

(payload) => {

console.*log*('Change received!', payload)

}

)

.*subscribe*()

Subscribe to updates

const channels = supabase.*channel*('custom-update-channel')

.*on*(

'postgres\_changes',

{ event: 'UPDATE', schema: 'public', table: 'etkinlikler' },

(payload) => {

console.*log*('Change received!', payload)

}

)

.*subscribe*()

Subscribe to deletes

const channels = supabase.*channel*('custom-delete-channel')

.*on*(

'postgres\_changes',

{ event: 'DELETE', schema: 'public', table: 'etkinlikler' },

(payload) => {

console.*log*('Change received!', payload)

}

)

.*subscribe*()

Subscribe to specific rows

const channels = supabase.*channel*('custom-filter-channel')

.*on*(

'postgres\_changes',

{ event: '\*', schema: 'public', table: 'etkinlikler', filter: 'some\_column=eq.some\_value' },

(payload) => {

console.*log*('Change received!', payload)

}

)

.*subscribe*()