

# Normalization and DB Creation

# Normalization

How you organize your data has a profound effect on how efficiently the data is stored, updated and selected.

Normalization is the process of:

- eliminating data redundancy
- enhancing data integrity
- eliminating unstructured data

Traditionally, there are 5 forms of normalization. Neatly named 1st, 2nd, 3rd, 4th, 5th and even 6<sup>th</sup> normal form.

In practice, only 1st through 3rd are typically used.

4<sup>th</sup>, 5<sup>th</sup> and 6th normal form are too restrictive for real use.

Even 3rd normal form is too restrictive, and therefore we will learn *Jeff Normal Form*, which is a less restrictive form of 3rd normal form.

In all normal forms each row of a table is uniquely identified by a primary key.

A primary key may be a single field or a combination of keys.

# Normalization – 1<sup>st</sup> Normal Form

1st Normal Form:

- breaks data into the smallest units possible
- data should be atomic
- each record must be unique

Observe the following table

People		
Name	Address	Phone
Jeff Salvage	1313 Mockingbird Lane, Mockingbird Heights, CA, 33004	(555) 555-1212
Ahsoka Tano	500 Republica Street, Coruscant City, Coruscant, 11111	(555) 101-1977
Thurman Munson	1977 Championship Lane, Bronx, NY 11002	(888)YAN-KEES

Therefore, any field that is non-atomic should be broken into separate fields.

People						
FirstName	LastName	Address	City	State	Zip	Phone
Jeff	Salvage	1313 Mockingbird Lane	Mockingbird Heights	CA	33004	(555) 555-1212
Ahsoka	Tano	500 Republica Street	Coruscant City	Coruscant	11111	(555) 101-1977
Thurman	Munson	1977 Championship Lane	Bronx	NY	11002	(888)YAN-KEES

Technically we should break up address, but we will leave that as is for now.

# Normalization – 1<sup>st</sup> Normal Form

Additionally, data should not be stored with repetitious groups of fields:

Observe the following table

ForceUsers		
Allegiance	ForceUser1	ForceUser2
Jedi	Qui Gon Jinn	Ahsoka Tano
Sith	Darth Maul	Count Dooko

When data is set up in this manner unused fields are wasted. In addition, a set number of values for the repeated field is hard coded into the table instead of allowing an arbitrary number of values.

Instead, you should create a record for each value as shown in the following table:

ForceAllegiance		
Allegiance	FirstName	LastName
Jedi	Qui Gon	Jinn
Jedi	Ahsoka	Tano
Sith	Darth	Maul
Sith	Count	Dooko

## Normalization – 1<sup>st</sup> Normal Form

However, now the table does not have a simple primary key.

Either we can add a field to the table to identify the force user or we can make the force user's first and last name part of the primary key.

ForceUser			
IDForceUser	Allegiance	FirstName	LastName
1	Jedi	Qui Gon	Jinn
2	Jedi	Ahsoka	Tano
3	Sith	Darth	Maul
4	Sith	Count	Dooko

# Normalization – 2<sup>nd</sup> Normal Form

Each progressive normal form builds on the previous normal form.

2<sup>nd</sup> Normal Form includes all of the rules that apply to 1<sup>st</sup> Normal Form.

2<sup>nd</sup> normal form states that in tables with compound primary keys, each non-key field should relate to a fact about all the keys (not a single part of the key) in the compound primary key.

Otherwise, the data should be reorganized into another table.

The following table is in 1<sup>st</sup> normal form and contains a compound primary key of *IDForceUser* and *IDAllegiance*.

ForceUser					
IDForceUser	IDAllegiance	Allegiance	FirstName	LastName	MidichlorianCount
1	1	Jedi	Qui Gon	Jinn	10,000
2	1	Jedi	Ahsoka	Tano	14,000
3	2	Sith	Darth	Maul	14,000
4	2	Sith	Count	Dooko	16,500

# Normalization – 2<sup>nd</sup> Normal Form

Observe the table broken into three tables that are in 2nd normal form.

ForceAttributes		
IDForceUser	IDAllegiance	MidichlorianCount
1	1	10,000
2	1	14,000
3	2	14,000
4	2	16,500

ForceUser		
IDForceUser	FirstName	LastName
1	Qui Gon	Jinn
2	Ahsoka	Tano
3	Darth	Maul
4	Count	Dooko

Allegiance	
IDAllegiance	Allegiance
1	Jedi
2	Sith

# Normalization – 3<sup>rd</sup> Normal Form

3rd normal form is similar to 2nd normal form.

The only difference is it applies to non-compound primary keys.

Each non key field should be a fact about the primary key.

Otherwise, it should be placed in a separate table.

Since the combination of *City*, *State* and *Zip* are repetitious, they do not belong in the table.

People							
<b>idPerson</b>	<b>FirstName</b>	<b>LastName</b>	<b>Address</b>	<b>City</b>	<b>State</b>	<b>Zip</b>	<b>Phone</b>
1	Jeff	Salvage	1313 Mockingbird Lane	Mockingbird Heights	CA	33004	(555) 555-1212
2	Ahsoka	Tano	500 Republica Street	Coruscant City	Coruscant	11111	(555) 101-1977
3	Thurman	Munson	1977 Championship Lane	Bronx	NY	11002	(888)YAN-KEES
4	Mon	Mothma	520 Republica Street	Coruscant City	Coruscant	11111	(555) 421-0608

*City*, *State*, and *Zip* should be in their own table.

# Normalization – 3<sup>rd</sup> Normal Form

The correct 3rd normal form is:

People					
idPerson	FirstName	LastName	Address	Zip	Phone
1	Jeff	Salvage	1313 Mockingbird Lane	33004	(555) 555-1212
2	Ahsoka	Tano	500 Republica Street	11111	(555) 101-1977
3	Thurman	Munson	1977 Championship Lane	11002	(888)YAN-KEES
4	Mon	Mothma	520 Republica Street	11111	(555) 421-0608

Address		
City	State	Zip
Mockingbird Heights	CA	33004
Coruscant City	Coruscant	11111
Bronx	NY	11002
Coruscant City	Coruscant	11111

# Normalization – Jeff Normal Form

Common sense.

Examples like the zip code take normalization too far.

Usually leave it as it was in 2nd normal form.

There are times it's ok not to normalize.

When the data is coming in as a unit as city, state, and zip are, that it's ok to leave it as is.

Also, zips do some strange things.

Not really a 1 to 1 relationship there.

# Normalization – Creating a Data Model

Let's create a data model for tracking international competitions in race walking.  
Specifically, we will focus on the judging of the event.

- Race walking is a progression of steps so taken that the walker makes contact with the ground so that no visible (to the human eye) loss of contact occurs.
- The advancing leg must be straightened (i.e., not bent at the knee) from the moment of first contact with the ground until in the vertical upright position.



# Normalization – Creating a Data Model

What is a "Bent Knee" in race walking:



# Normalization – Creating a Data Model

What is a "Loss of Contact" in race walking:



# Normalization – Creating a Data Model

All the events that occur in the race are tracking with a Summary Sheet  
 Our job is to develop a data model for the race

JUDGING SUMMARY SHEET (RACE WALKING)																				
DATE			START TIME			EVENT								CHIEF JUDGES NAME						
24th August 2023			7:00			World Athletics Championships, Budapest 2023- 35km Race Walk Women								Zoë Eastwood-Bryson (AUS)						
Judge's Name	Jean-Pierre DAHM FRA	José DIAS POR	Anne FRÖBERG FIN	Daniel MICHAUD CAN	Guillermo PERA VALLEOS ARG	Ian RICHARDS GBR	Sergio SOLANA SOR ESP	Man chun YEUNG HKG	Penalty Zone		Chief Judge	DQ Notification	Time	Check of	Yellow Paddles & Disqualifications	Yellow Paddles & Disqualifications				
Number	1	2	3	4	5	6	7	8	Athlete's Number	Yellow Paddle RC	Yellow Paddle RC	Yellow Paddle RC	Yellow Paddle RC	Yellow Paddle RC	Yellow Paddle RC	Yellow Paddle RC	Entry Time Exit Time	Time Offence	Time	~ < RC
159									ORTEGA Alejandra MEX 2:50:44 (12.)									1 1 0		
163									GARCIA LEÓN Kimberly PER 2:40:52 (2.)									0 2 1		
164									INGA Evelyn PER 2:46:18 (6.)									3 0 0		
165									CHOJECKA Olga POL 2:46:48 (8.)									1 1 1		
166									ELLWARD Agnieszka POL 2:56:51 (18.)									1 1 0		
167									ZDZIEBŁO Katarzyna POL DQ 54.7.5									2 0 0		
169									HENRIQUES Inês POR DNF									0 0 0		
172									ACATRINEI Mihaela ROU DNF									0 0 0		
173									RODEAN Ana Veronica ROU DNF									0 0 0		
174									BURZALOVÁ Hana SVK 3:02:47 (28.)									0 1 0		
CHECK	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	PAGE	1 2 0 2 0 1 2 4 2 0 2 0 0 2 1 2 1 1 3 0 1 1 0 0							~ < RC			
TOTAL	3 4 1 7 2 2 5 8 6 4 4 0 1 6 3 3 3 3 7 5 3 3 1 1 0 31 33 18																~ < RC			
ASSISTANTS TO CHIEF JUDGE NAME												RECORDER'S NAME								
Orsolya Gruber (HUN)												Zuzana Costin (SVK)								

# Normalization – Creating a Data Model

What attributes define a race?

JUDGING SUMMARY SHEET (RACE WALKING)																	
DATE		START TIME		EVENT								CHIEF JUDGES NAME					
24th August 2023		7:00		World Athletics Championships, Budapest 2023- 35km Race Walk Women								Zoë Eastwood-Bryson (AUS)					
Judge's Name	Jean-Pierre DAHM FRA	José DIAS POR	Anne FROBERG FIN	Daniel MICHAUD CAN	Guillermo PERA VALLEOS ARG	Ian RICHARDS GBR	Sergio SOLANA SOR ESP	Man chun YEUNG HKG	Penalty Zone	Chief Judge	DQ Notification	Check of Yellow Paddles & Disqualifications					
Number	1	2	3	4	5	6	7	8									
Athlete's Number	Yellow Paddle ~ < RC	Yellow Paddle ~ < RC	Yellow Paddle ~ < RC	Yellow Paddle ~ < RC	Yellow Paddle ~ < RC	Yellow Paddle ~ < RC	Yellow Paddle ~ < RC	Yellow Paddle ~ < RC	Entry Time	Time Offence	Time	~ < RC					
									Exit Time								
159	ORTEGA Alejandra MEX 2:50:44 (12.)												1	1	0		
163	GARCIA LEÓN Kimberly PER 2:40:52 (2.)												0	2	1		
164	INGA Evelyn PER 2:46:18 (6.)												3	0	0		
165	CHOJECKA Olga POL 2:46:48 (8.)												1	1	1		
166	ELLWARD Agnieszka POL 2:56:51 (18.)												1	1	0		
167	ZDZIEBŁO Katarzyna POL DQ 54.7.5												8:54	9:27	3	2	4
169	HENRIQUES Inês POR DNF												8:57				
172	ACATRINEI Mihaela ROU DNF												0	3	0		
173	RODEAN Ana Veronica ROU DNF												0	0	0		
174	BURZALOVÁ Hana SVK 3:02:47 (28.)												0	1	0		
CHECK	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC					~ < RC				
PAGE	1 2 0	2 0 1	2 4 2	0 2 0	2 0 0	2 0 0	2 1 1	1 2 1	1 1 3	0 1 1	0 0 0		11 11 6				
CHECK	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC					~ < RC				
TOTAL	3 4 1	7 2 2	5 8 6	4 4 0	1 6 3	3 3 3	7 5 3	1 1 0	31 33 18				31 33 18				
ASSISTANTS TO CHIEF JUDGE NAME												RECODER'S NAME					
Orsolya Gruber (HUN)												Zuzana Costin (SVK)					

# Normalization – Creating a Data Model

## What attributes define a race?

The event field on the spreadsheet is a compound field.

It contains:

# The event name: World Athletics Championships

City: Budapest

Year: 2023, repeated elsewhere

Distance: 35km

## Gender: Women

RaceDate: 8/24/2023

**Start Time: 7:00**

**Country:** Not listed, but should be

This informs the creation of:

Race								
IDRace	Event	City	Country	RaceDate	StartTime	Distance	DistanceUnits	Gender
1	World Athletics Championships	Budapest	Hungary	2023-08-24	7:00 AM	35	km	F
2	World Athletics Championships	Budapest	Hungary	2023-08-24	7:00 AM	35	km	M
3	World Athletics Championships	Budapest	Hungary	2023-08-20	7:15 AM	20	km	F
4	World Athletics Championships	Budapest	Hungary	2023-08-19	10:50 AM	20	km	M

# Normalization – Creating a Data Model

We already have issues of repetitious data related to event, city and country.

This informs the creation of:

Event			
IDEvent	Event	City	Country
1	World Athletics Championships	Budapest	Hungary
2	World Athletics Team Championships	Anatalya	Turkey
3	Olympics Games	Paris	France

Race						
IDRace	IDEvent	RaceDate	StartTime	Distance	DistanceUnits	Gender
1	1	2023-08-24	7:00 AM	35	km	F
2	1	2023-08-24	7:00 AM	35	km	M
3	1	2023-08-20	7:15 AM	20	km	F
4	1	2023-08-19	10:50 AM	20	km	M

# Normalization – Creating a Data Model

Let's create a relation for Athletes

We need to track:

FirstName: Alejandra

LastName: ORTEGA

Country: Mex

BibNumber: 159

FinishingTime: 2:50:44

FinishingPlace: 12

		DATE		START TIME		EVENT								CHIEF JUDGES NAME												
		24th	August	2023	7:00	World Athletics Championships, Budapest 2023- 35km Race Walk Women								Zoë Eastwood-Bryson (AUS)												
Judge's Name		Jean-Pierre DAHM FRA	José DIAS POR	Anne FRÖBERG FIN	Daniel MICHAUD CAN	Guillermo PERA VALLEOS ARG	Ian RICHARDS GBR	Sergio SOLANA SOR ESP	Man chun YEUNG HKG	Penalty Zone	Chief Judge	DQ Notification	Check of Yellow Paddles & Disqualifications													
Number		1	2	3	4	5	6	7	8					Entry Time	Time Offence	Time										
Athlete's Number		Yellow Paddle ~ <	RC	Yellow Paddle ~ <	RC	Yellow Paddle ~ <	RC	Yellow Paddle ~ <	RC	Yellow Paddle ~ <	RC	Yellow Paddle ~ <	RC	Yellow Paddle ~ <	RC	Yellow Paddle ~ <	RC									
159		ORTEGA Alejandra MEX 2:50:44 (12.)								9:21	9:37	9:23	9:23	9:23	9:23	9:23	9:23	9:23								
163		GARCIA LEÓN Kimberly PER 2:40:52 (2.)								8:49	8:47	9:15	9:15	9:15	9:15	9:15	9:15	9:15								
164		INGA Evelyn PER 2:46:18 (6.)								8:35	8:23	9:32	9:32	9:32	9:32	9:32	9:32	9:32								
165		CHOJECKA Olga POL 2:46:48 (8.)								7:31	8:10	9:26	9:26	9:26	9:26	9:26	9:26	9:26								
166		ELLWARD Agnieszka POL 2:56:51 (18.)								9:25	9:25	9:25	9:25	9:25	9:25	9:25	9:25	9:25								
167		ZDZIEBŁO Katarzyna POL DQ 54.7.5								9:09	7:29	7:58	9:15	9:26	8:21	8:45	8:24	8:54								
169		HENRIQUES Inês POR DNF								~	7:12	7:30	7:30	7:30	7:30	7:30	7:30	7:30								
172		ACATRINEI Mihaela ROU DNF								7:15	7:26	7:18	7:18	7:18	7:18	7:18	7:18	7:18								
173		RODEAN Ana Veronica ROU DNF								8:48	8:48	8:48	8:48	8:48	8:48	8:48	8:48	8:48								
174		BURZALOVÁ Hana SVK 3:02:47 (28.)								8:48	8:48	8:48	8:48	8:48	8:48	8:48	8:48	8:48								
CHECK		~ <	RC	~ <	RC	~ <	RC	~ <	RC	~ <	RC	~ <	RC	~ <	RC	~ <	RC									
PAGE		1	2	0	2	0	1	2	4	2	0	2	0	2	1	1	3	0								
CHECK		~ <	RC	~ <	RC	~ <	RC	~ <	RC	~ <	RC	~ <	RC	~ <	RC	~ <	RC									
TOTAL		3	4	1	7	2	2	5	8	6	4	4	0	1	6	3	3	7	5	3	1	1	0	31	33	18
ASSISTANTS TO CHIEF JUDGE NAME												RECODER'S NAME				Zuzana Costin (SVK)										
Orsolya Gruber (HUN)																										

# Normalization – Creating a Data Model

# Let's create a relation for Athletes

## We need to track:

FirstName: Alejandra

Last Name: ORTEGA

Country: Mex

BibNumber: 159

FinishingTime: 2:50:44

FinishingPlace: 12

Finished: Yes / DNF / DQ

This informs the creation of:

Race							
<b>idAthlete</b>	<b>FirstName</b>	<b>LastName</b>	<b>CountryCode</b>	<b>BibNumber</b>	<b>FinishingTime</b>	<b>FinishingPlace</b>	<b>Finished</b>
1	Alejandra	ORTEGA	MEX	159	2:50:44	12	Yes
2	Kimberly	GARCÍA LEÓN	PER	163	2:40:52	2	Yes
3	Evelyn	INGA	PER	164	2:46:18	6	Yes
4	Olga	CHOKECKA	POL	165	2:46:48	20	Yes

# Normalization – Creating a Data Model

Again, we have issues related to *BibNumber*, *FinishingTime*, *FinishingPlace*

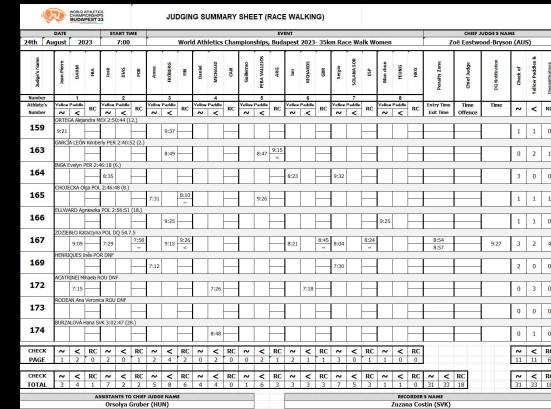
Athletes will compete in more than one race/event.

Race specific data must be separated from athlete specific data

Refactor as follows:

Athlete			
IDAthlete	FirstName	LastName	CountryCode
1	Alejandra	ORTEGA	MEX
2	Kimberly	GARCÍA LEÓN	PER
3	Evelyn	INGA	PER
4	Olga	CHOKECKA	POL

IDRace	IDAthlete	BibNumber	FinishingTime	FinishingPlace	Finished
1	1	159	2:50:44	12	Yes
1	2	163	2:40:52	2	Yes
1	3	164	2:46:18	6	Yes
1	4	165	2:46:48	20	Yes



# Normalization – Creating a Data Model

Judges need to be tracked:

Judge			
IDJudge	FirstName	LastName	CountryCode
1	Jean Pierre	DAHM	FRA
2	José	DIAS	PER
3	Anne	FRÖBERG	FIN
4	Daniel	MICHAUD	CAN

JUDGING SUMMARY SHEET (RACE WALKING)												CHIEF JUDGES NAME			
DATE			START TIME			EVENT						CHIEF JUDGES NAME			
24th August 2023			7:00			World Athletics Championships, Budapest 2023- 35km Race Walk Women						Zoë Eastwood-Bryson (AUS)			
Judge's Name	Jean-Pierre DAHM FRA	José DIAS POR	Anne FRÖBERG FIN	Daniel MICHAUD CAN	Guillermo PERA VALLEOS ARG	Ian RICHARDS GBR	Sergio SOLANA SOR ESP	Man chun YEUNG HKG	Penalty Zone	Chief Judge	DQ Notification	Check of Yellow Paddles & Disqualifications			
Number	1	2	3	4	5	6	7	8							
Athlete's Number	Yellow Paddle ~ < RC	Yellow Paddle ~ < RC	Yellow Paddle ~ < RC	Yellow Paddle ~ < RC	Yellow Paddle ~ < RC	Yellow Paddle ~ < RC	Yellow Paddle ~ < RC	Yellow Paddle ~ < RC	Entry Time	Time Offence	Time	~	~	RC	
	ORTEGA Alejandra MEX 2:50:44 (12.)								Exit Time			1	1	0	
159	9:21		9:37												
163	GARCIA LEÓN Kimberly PER 2:40:52 (2.)			8:49		8:47	9:15						0	2	1
164	INGA Evelyn PER 2:46:18 (6.)				8:23		9:32						3	0	0
165	CHOJECKA Olga POL 2:46:48 (8.)		7:31	8:10		9:26							1	1	1
166	ELLWARD Agnieszka POL 2:56:51 (18.)			9:25				9:25					1	1	0
167	ZDZIEBŁO Katarzyna POL DQ 54.7.5	9:09	7:29	7:58	9:15	9:26		8:21	8:45	8:04	8:24		8:54	8:57	9:27
169	HENRIQUES Inês POR DNF			7:12				7:30					2	0	0
172	ACATRINEI Mihaela ROU DNF	7:15			7:26		7:18						0	3	0
173	RODEAN Ana Veronica ROU DNF												0	0	0
174	BURZALOVÁ Hana SVK 3:02:47 (28.)			8:48									0	1	0
CHECK	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	
PAGE	1 2 0	2 0 1	2 4 2	0 2 0	0 2 0	0 2 1	1 2 1	1 3 0	1 1 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0
CHECK	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	~ < RC	
TOTAL	3 4 1	7 2 2	5 8 6	4 4 0	0 1 6	3 3 3	3 3 7	5 3 3	1 1 0	31 33 18			31	33	18
ASSISTANTS TO CHIEF JUDGE NAME												RECODER'S NAME			
Orsolya Gruber (HUN)												Zuzana Costin (SVK)			

This is not enough.

Judges work more than one event/race

A table is needed to connect the judge to the race.

RaceJudge	
IDRace	IDJudge
1	1
2	2
3	3
4	4

# Normalization – Creating a Data Model

Judges can take no action.

Judges can make calls.

When a judge makes a call, it can in one of four variations, but they can only make each call once per athlete.

Yellow (caution) for < (bent knee)

Yellow (caution) for ~ (loss of contact)

Red (proposal for disqualification) for < (bent knee)

Red (proposal for disqualification) for ~ (loss of contact)

When a judge makes a call, they also record:

- the time of day
- the bib number of the athlete
- their judge number
- the race

We can store it in a relation like:

JudgeCall						
IDRace	IDJudge	Color	Infraction	TOD	BibNumber	
1	1	Yellow	~	9:33 AM	181	
1	2	Yellow	~	8:04 AM	142	
1	4	Yellow	<	8:20 AM	183	
1	1	Red	~	08:10 AM	181	

# Normalization – Creating a Data Model

Technically, `Yellow` and `Red` could be moved out to a reference table and IDs used instead.  
`Red` and `Yellow` are not likely to change, they are small text fields, and make readability easier.  
Either way would be fine.

JudgeCall					
IDRace	IDJudge	IDColor	Infraction	TOD	BibNumber
1	1	1	~	9:33 AM	181
1	2	1	~	8:04 AM	142
1	4	1	<	8:20 AM	183
1	1	2	~	08:10 AM	181

Color	
IDColor	Color
1	Yellow
2	Red

# Normalization – Creating a Data Model

Race Walking Data Model:

Event					Race								
idEvent	Event		City	Country	idRace	IDEvent	RaceDate	StartTime	Distance	DistanceUnits	Gender		
1	World Athletics Championships		Budapest	Hungary	1	1	2023-08-24	7:00 AM	35	km	F		
2	World Athletics Team Championships		Anatalya	Turkey	2	1	2023-08-24	7:00 AM	35	km	M		
3	Olympics Games		Paris	France	3	1	2023-08-20	7:15 AM	20	km	F		
Athlete					4	1	2023-08-19	10:50 AM	20	km	M		
Athlete					Bib								
idAthlete	FirstName	LastName	CountryCode	Gender	IDRace	IDAthlete	BibNumber	FinishingTime	FinishingPlace	Finished			
1	Alejandra	ORTEGA	MEX	F	1	1	159	2:50:44	12	Yes			
2	Kimberly	GARCÍA LEÓN	PER	F	1	2	163	2:40:52	2	Yes			
3	Evelyn	INGA	PER	F	1	3	164	2:46:18	6	Yes			
4	Olga	CHOKECKA	POL	F	1	4	165	2:46:48	20	Yes			
Judge				RaceJudge		JudgeCall							
IDJudge	FirstName	LastName	CountryCode	IDRace	IDJudge	idRace	IDJudge	Color	Infraction	TOD	BibNumber		
1	Jean Pierre	DAHM	FRA	1	1	1	1	Yellow	~	9:33 AM	181		
2	José	DIAS	PER	2	2	1	2	Yellow	~	8:04 AM	142		
3	Anne	FRÖBERG	FIN	3	3	1	4	Yellow	<	8:20 AM	183		
4	Daniel	MICHAUD	CAN	4	4	1	1	Red	~	08:10 AM	181		

We have more work to do, let's identify the primary keys of each table by underlining the fields that comprise the primary key

# Normalization – Creating a Data Model

Race Walking Data Model:

Event				Race										
<u>IDEvent</u>	Event		City	Country	<u>IDRace</u>	<u>IDEvent</u>	RaceDate	StartTime	Distance	DistanceUnits	Gender			
1	World Athletics Championships			Budapest	Hungary	1	1	8/24/2023	7:00 AM	35	km	F		
2	World Athletics Team Championships			Anatalya	Turkey	2	1	8/24/2023	7:00 AM	35	km	M		
3	Olympics Games			Paris	France	3	1	8/20/2023	7:15 AM	20	km	F		
Athlete					4	1	8/19/2023	10:50 AM	20	km	M			
Bib														
<u>IDRace</u>	<u>IDAthlete</u>	<u>BibNumber</u>		FinishingTime	FinishingPlace		Finished							
1	1	159		2:50:44	12		Yes							
1	2	163		2:40:52	2		Yes							
1	3	164		2:46:18	6		Yes							
1	4	165		2:46:48	20		Yes							
Judge				RaceJudge		JudgeCall								
<u>IDJudge</u>	FirstName	LastName	CountryCode	<u>IDRace</u>	<u>IDJudge</u>	<u>IDRace</u>	<u>IDJudge</u>	Color	Infraction	<u>TOD</u>	<u>BibNumber</u>			
1	Jean Pierre	DAHM	FRA	1	1	1	1	Yellow	~	9:33 AM	181			
2	José	DIAS	PER	2	2	1	2	Yellow	~	8:04 AM	142			
3	Anne	FRÖBERG	FIN	3	3	1	4	Yellow	<	8:20 AM	183			
4	Daniel	MICHAUD	CAN	4	4	1	1	Red	~	08:10 AM	181			

We have more work to do, let's identify the primary keys of each table by underlining the fields that comprise the primary key

# Normalization – Creating a Data Model

We need two more tables to store manual visual video observations made about the race

Create an *Observer* table to store the people making the observations with a Primary Key of *IDObserver*:

Observer		
<u>IDObserver</u>	FirstName	LastName
1	Jeff	Salvage
2	David	Harriman
3	Gary	Westerfield

Create an *VideoObservation* table to store observations of the race walkers with a Primary Key of *ID*:

VideoObservation												
<u>ID</u>	IDRace	BibNumber	IDObserver	LOCAverage	LOC1	LOC2	LOC3	LOC4	BentKneeAngle	Comment	VideoFile	TOD
1	1	159	2	46.9	5	6	5.5	6	0		C1002	09:10 AM
2	1	163	1		6	6	6	6.5	0			
3	1	164	1						176.4			

# Normalization – Installing SQLite & DBVisualizer

"Install" SQLite

<https://www.sqlite.org/download.html>

Download the file for your OS containing the following:

A bundle of command-line tools for managing SQLite database files, including the command-line shell program, the sqldiff.exe program, and the sqlite3\_analyzer.exe program. 64-bit.

Unzip it.

Download and install DBVisualizer:

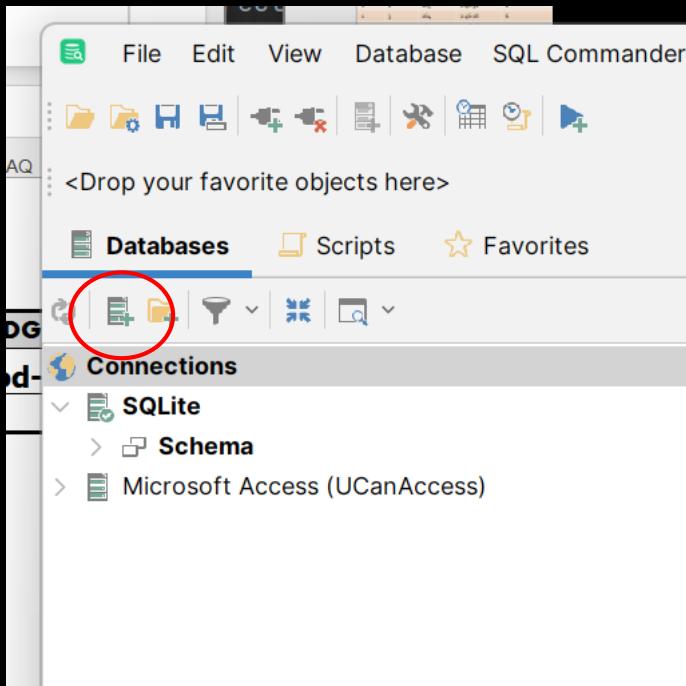
<https://www.dbvis.com/download/>

Use the license located in Learn.

Please do not share it.

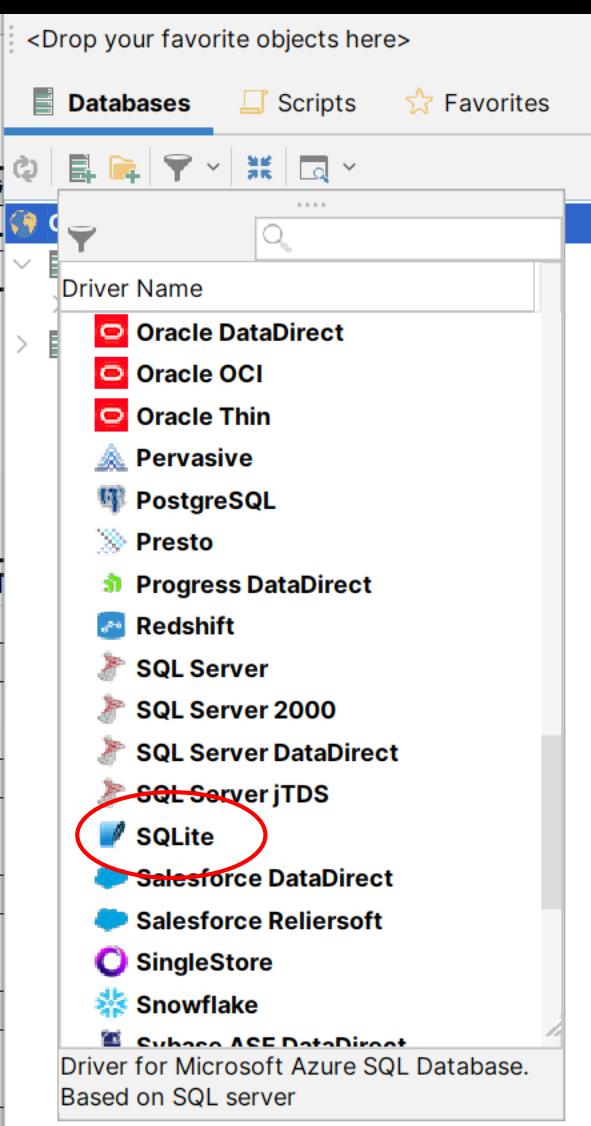
# Normalization – Creating a Database Connection

Click on the icon to create a new database connection:



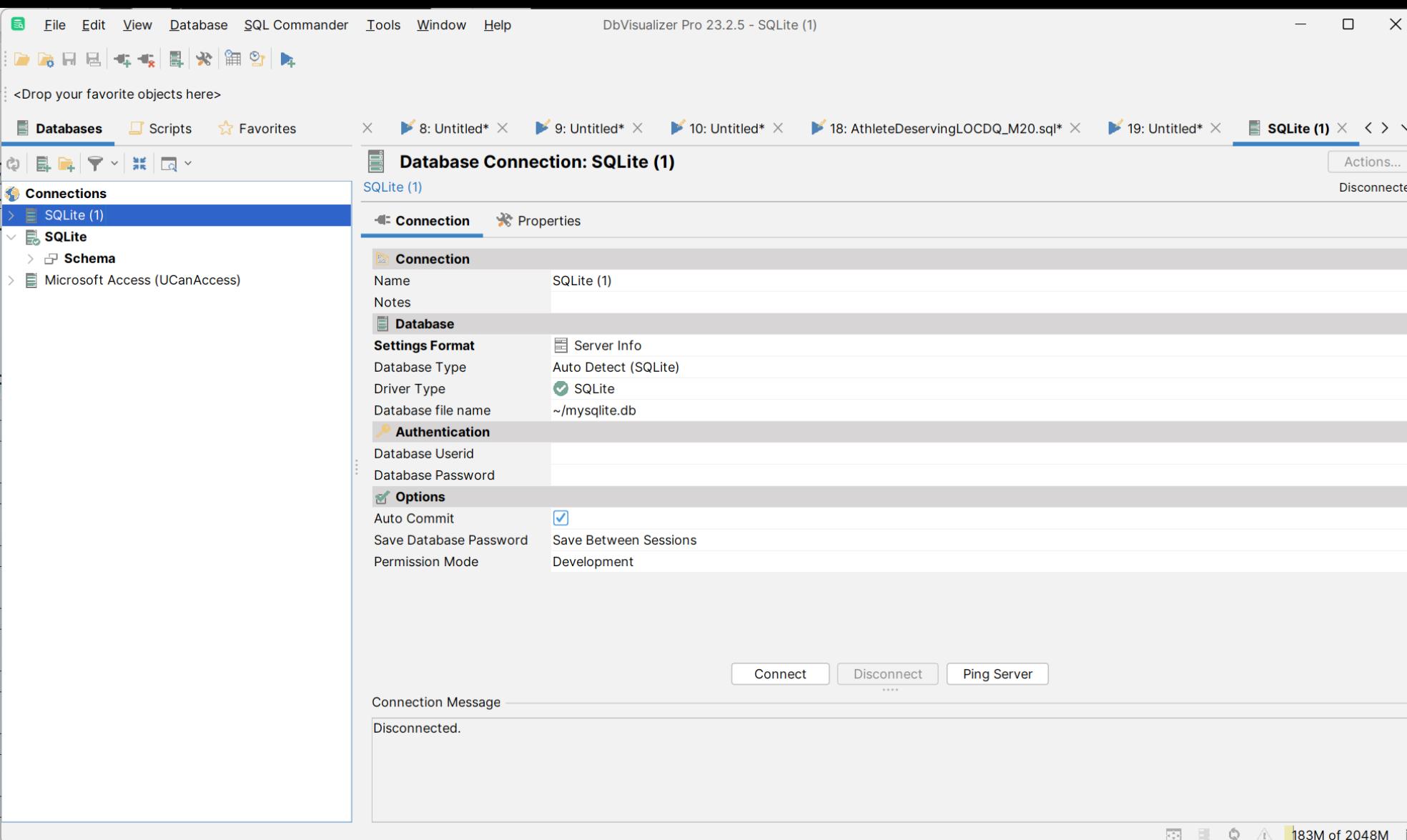
# Normalization – Creating a Database Connection

Select *SQLite* from the *Drive Name* pulldown:



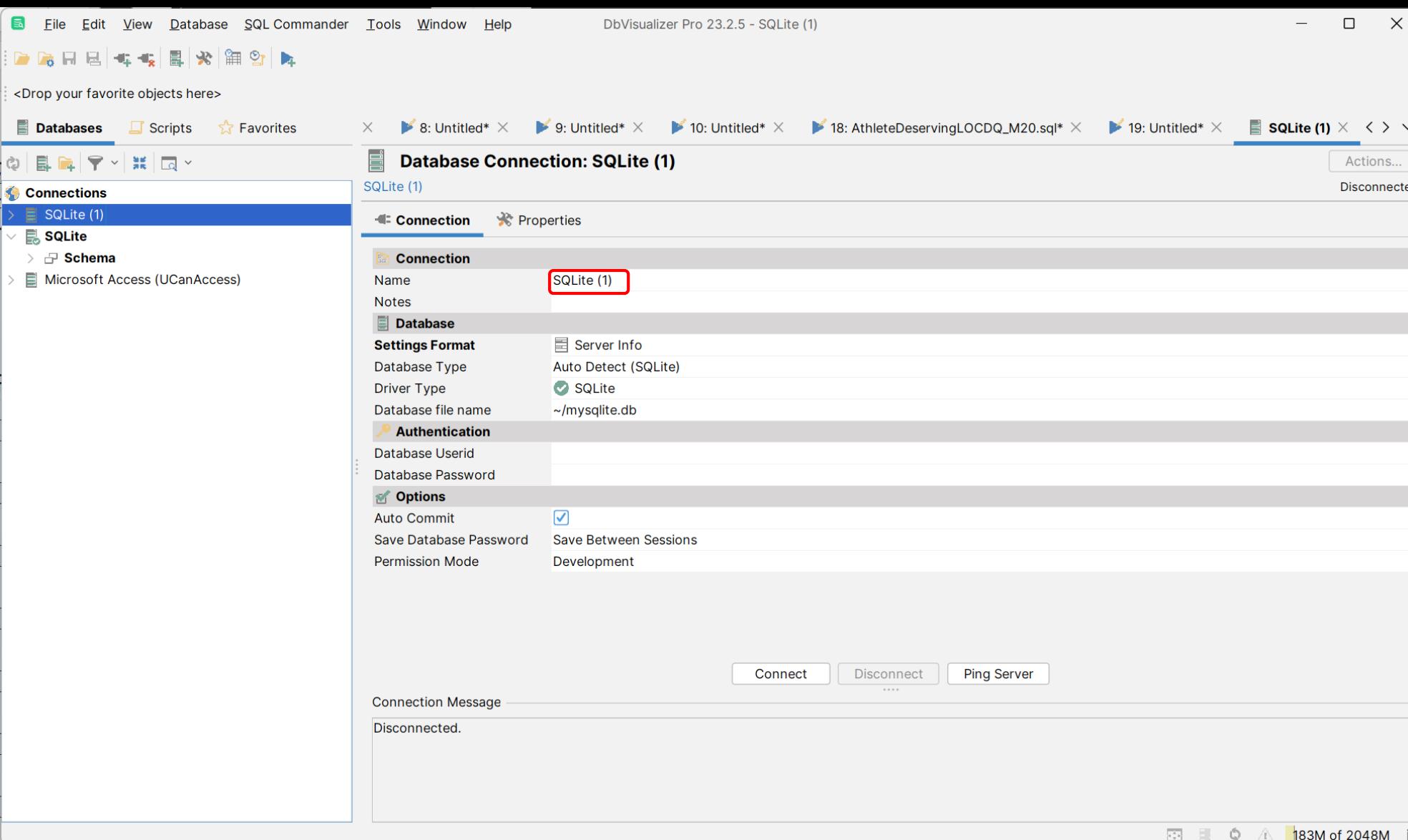
# Normalization – Creating a Database Connection

A new *Connection* dialog is presented:



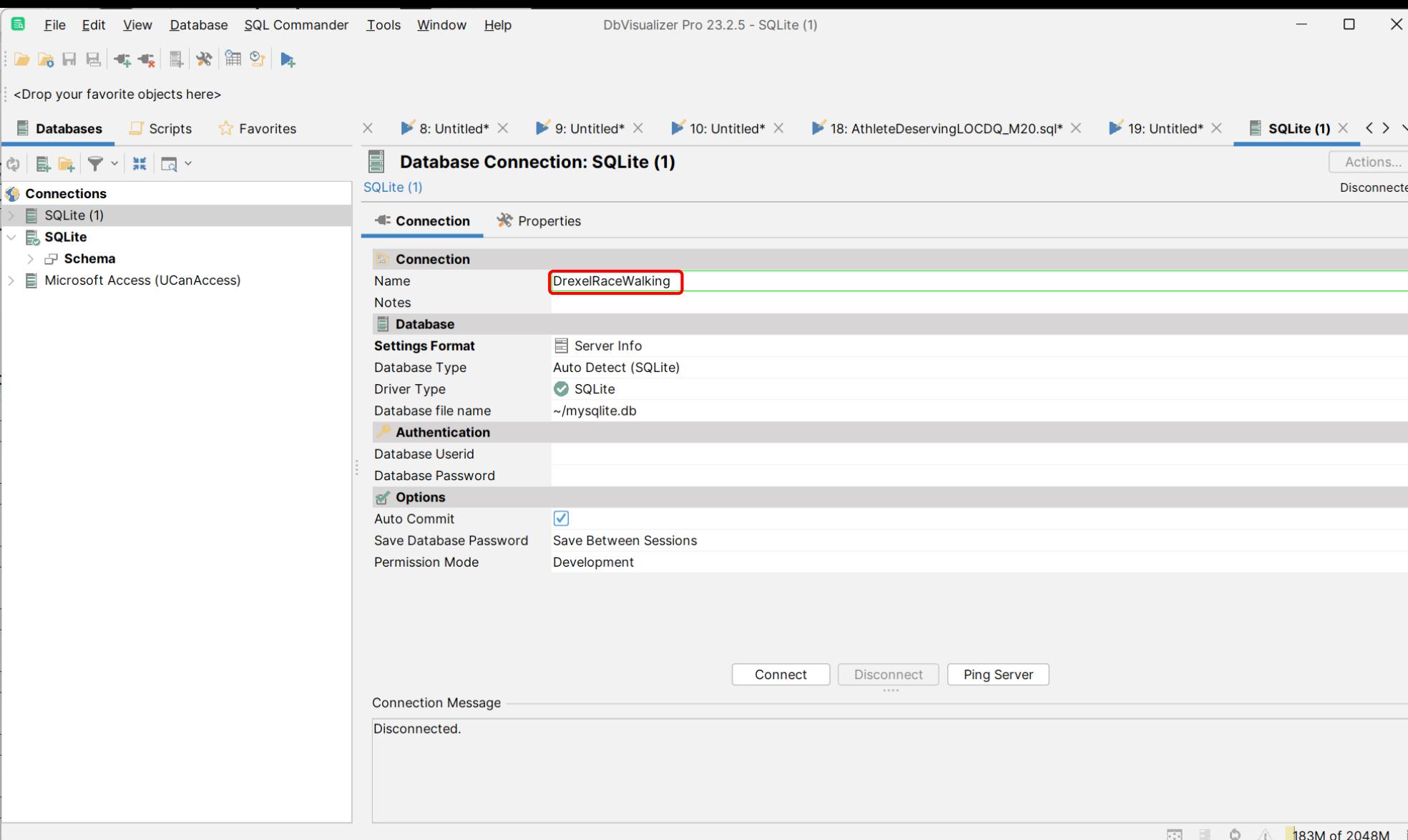
# Normalization – Creating a Database Connection

Change the *Name* of the connection to something appropriate like *DrexelRaceWalking*



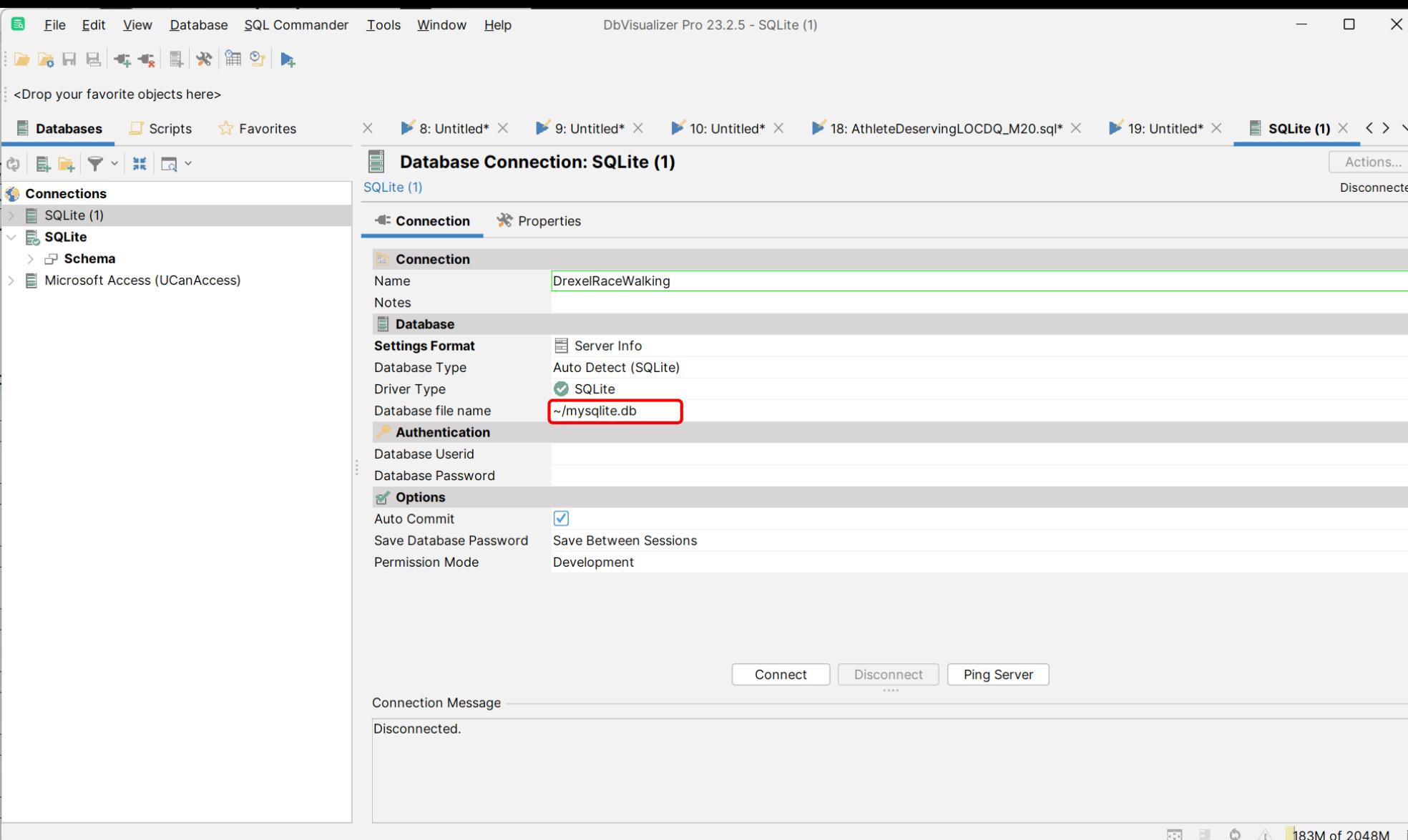
# Normalization – Creating a Database Connection

Change the *Name* of the connection to something appropriate like *DrexelRaceWalking*



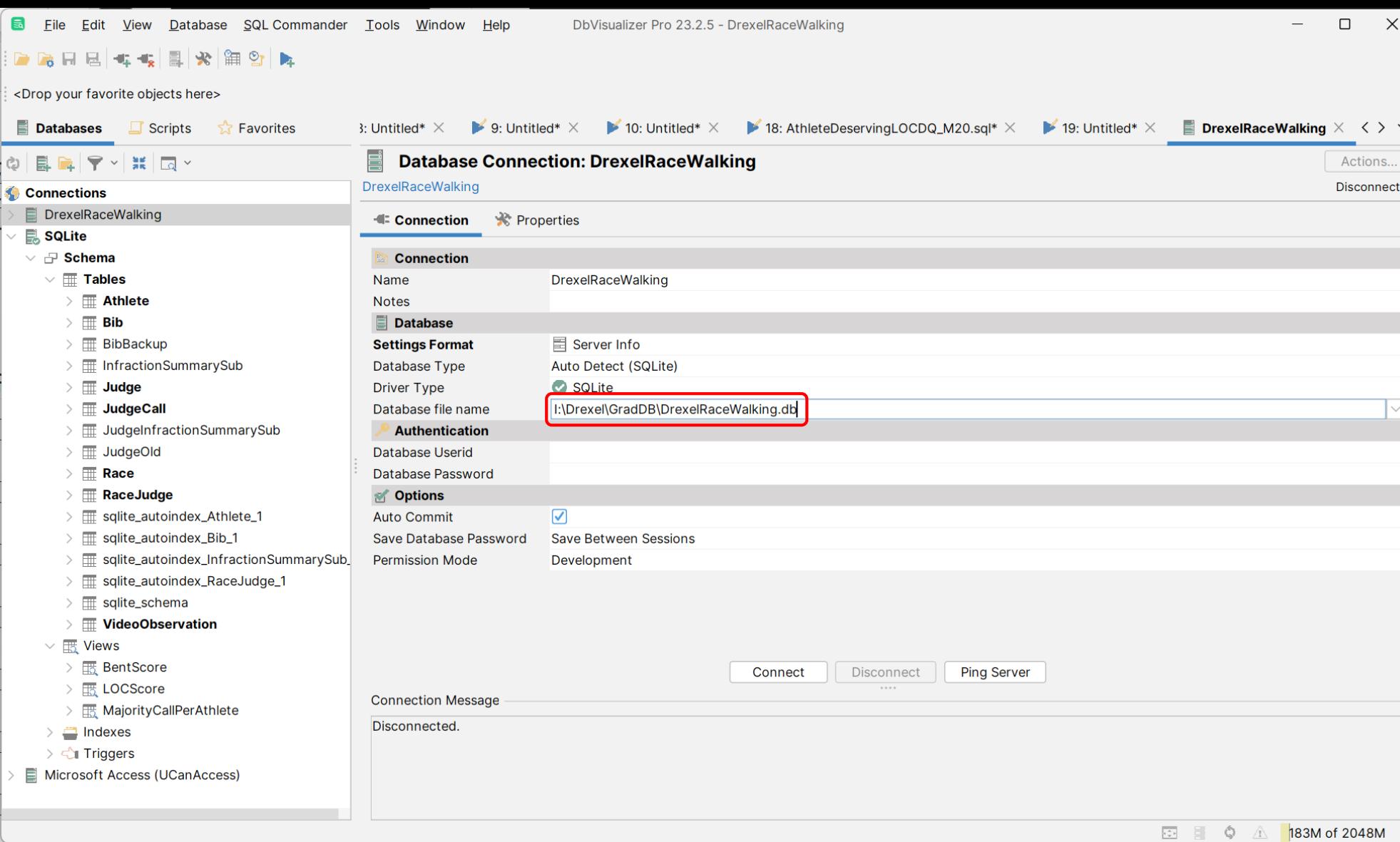
# Normalization – Creating a Database Connection

Change the location / name of the physical file the database is stored within:



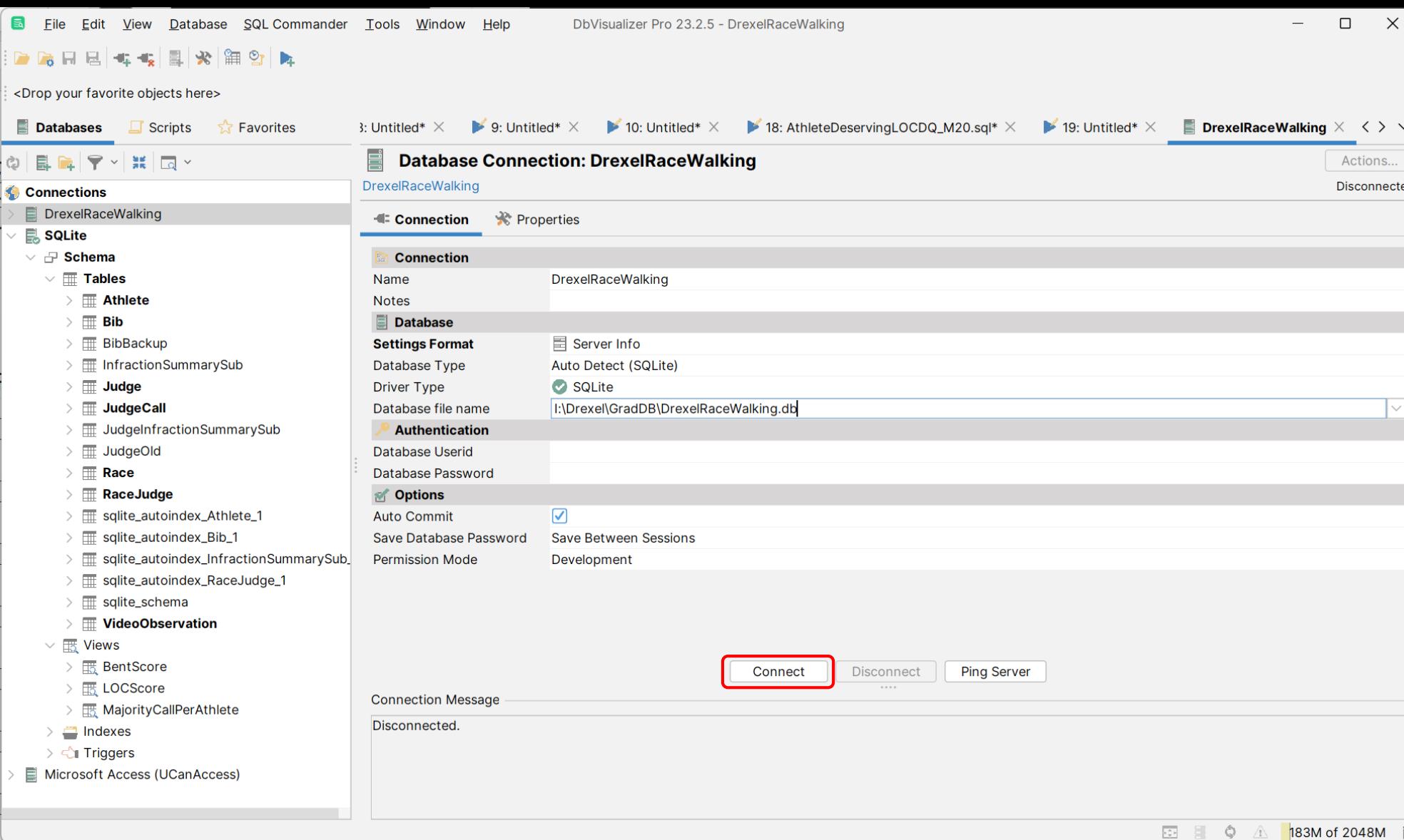
# Normalization – Creating a Database Connection

Change the location / name of the physical file the database is stored within:



# Normalization – Creating a Database Connection

Click on the *Connect* button:



# Normalization – Creating a Database Connection

Your new database appears and can be expanded on the left:

The screenshot shows the DbVisualizer Pro 23.2.5 interface. The title bar reads "DbVisualizer Pro 23.2.5 - DrexelRaceWalking". The menu bar includes File, Edit, View, Database, SQL Commander, Tools, Window, and Help. The toolbar has icons for file operations like Open, Save, and Print. A message bar at the top says "<Drop your favorite objects here>". The main window has tabs for Databases, Scripts, and Favorites. The Databases tab is selected, showing a list of connections: "DrexelRaceWalking" (selected), "SQLite", and "Microsoft Access (UCanAccess)". The "DrexelRaceWalking" connection is expanded, showing its schema structure with nodes for Schema, Tables, Views, Indexes, and Triggers. A red oval highlights this expanded schema area. The central workspace is titled "Database Connection: DrexelRaceWalking" with the URL "jdbc:sqlite:I:\Drexel\GradDB\DrexelRaceWalking.db". It displays the "Connection" tab with the following settings:

Name	DrexelRaceWalking
Notes	
Database	
Settings Format	Server Info
Database Type	SQLite
Driver Type	SQLite
Database file name	I:\Drexel\GradDB\DrexelRaceWalking.db
Authentication	
Database Userid	
Database Password	
Options	
Auto Commit	<input checked="" type="checkbox"/>
Save Database Password	<input type="checkbox"/>
Permission Mode	Development

At the bottom of the workspace, there are buttons for Reconnect, Disconnect, and Ping Server. Below the workspace is a "Connection Message" panel containing the text:

```
SQLite  
3.41.2  
SQLite JDBC  
3.41.2.1
```

The status bar at the bottom right shows "183M of 2048M".

Currently, there are no tables.

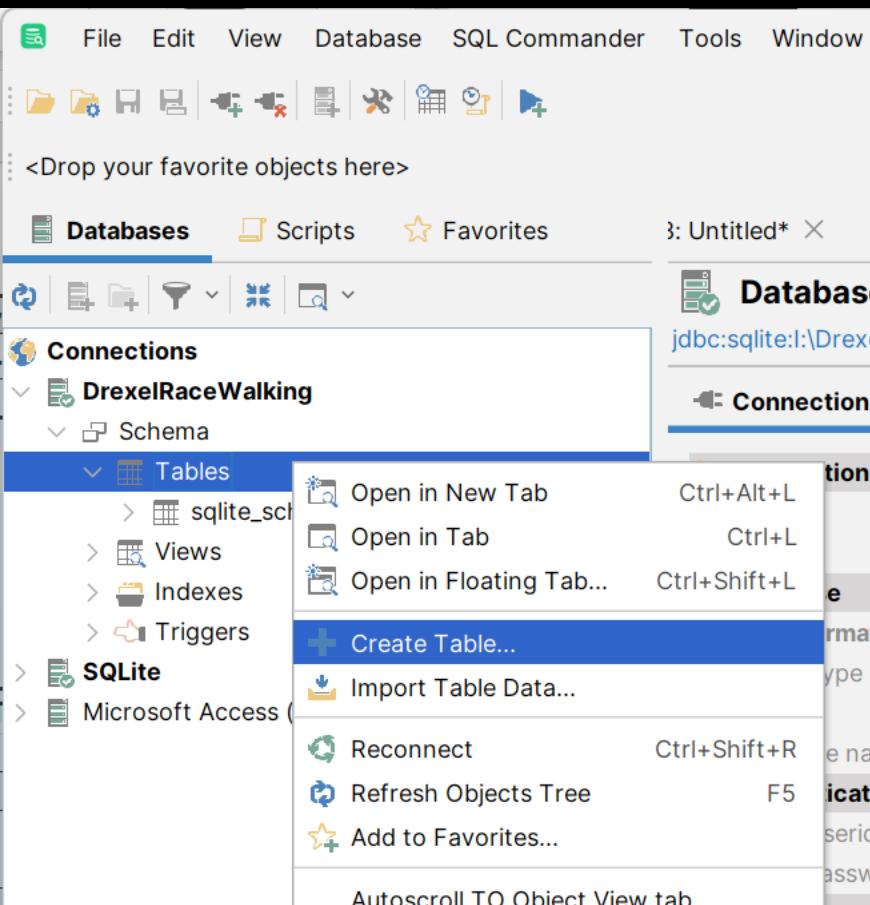
# Creating Tables - GUI

We can create tables with the GUI or by SQL.

Use the GUI.

Right Click on *Tables*.

Select *Create Table...* from the popup:



Event			
idEvent	Event	City	Country
1	World Athletics Championships	Budapest	Hungary
2	World Athletics Team Championships	Anatalya	Turkey
3	Olympics Games	Paris	France

# Creating Tables - GUI

Enter the table name, *Event*:

Create Table - DrexelRaceWalking/Schema/Tables X

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table:

**Columns** Primary Key Foreign Keys Unique Constraints Check Constraints

Name	Data Type	Size	Scale	Nullable	Default
Result set is empty					

+  -  ▲  ▼

Show SQL Execute Cancel

# Creating Tables - GUI

Add the columns / fields, one at a time, by clicking on the + icon:

Create Table - DrexelRaceWalking/Schema/Tables X

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: Event

**Columns** Primary Key Foreign Keys Unique Constraints Check Constraints

Name	Data Type	Size	Scale	Nullable	Default	<span style="color: red; border: 1px solid red; padding: 2px;">+</span>
Result set is empty						

Show SQL Execute Cancel

The screenshot shows a 'Create Table' dialog box from a database management tool. The 'Table' field is populated with 'Event'. Below the table definition is a table for defining columns. The 'Default' column header is underlined in blue. A red box highlights the '+' icon in the 'Default' column, indicating it's the target for adding new columns. The 'Show SQL' checkbox is checked. At the bottom are 'Execute' and 'Cancel' buttons.

# Creating Tables - GUI

Add the *IDEvent* field, after clicking on the **+**, default values appear:

Create Table - DrexelRaceWalking/Schema/Tables X

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

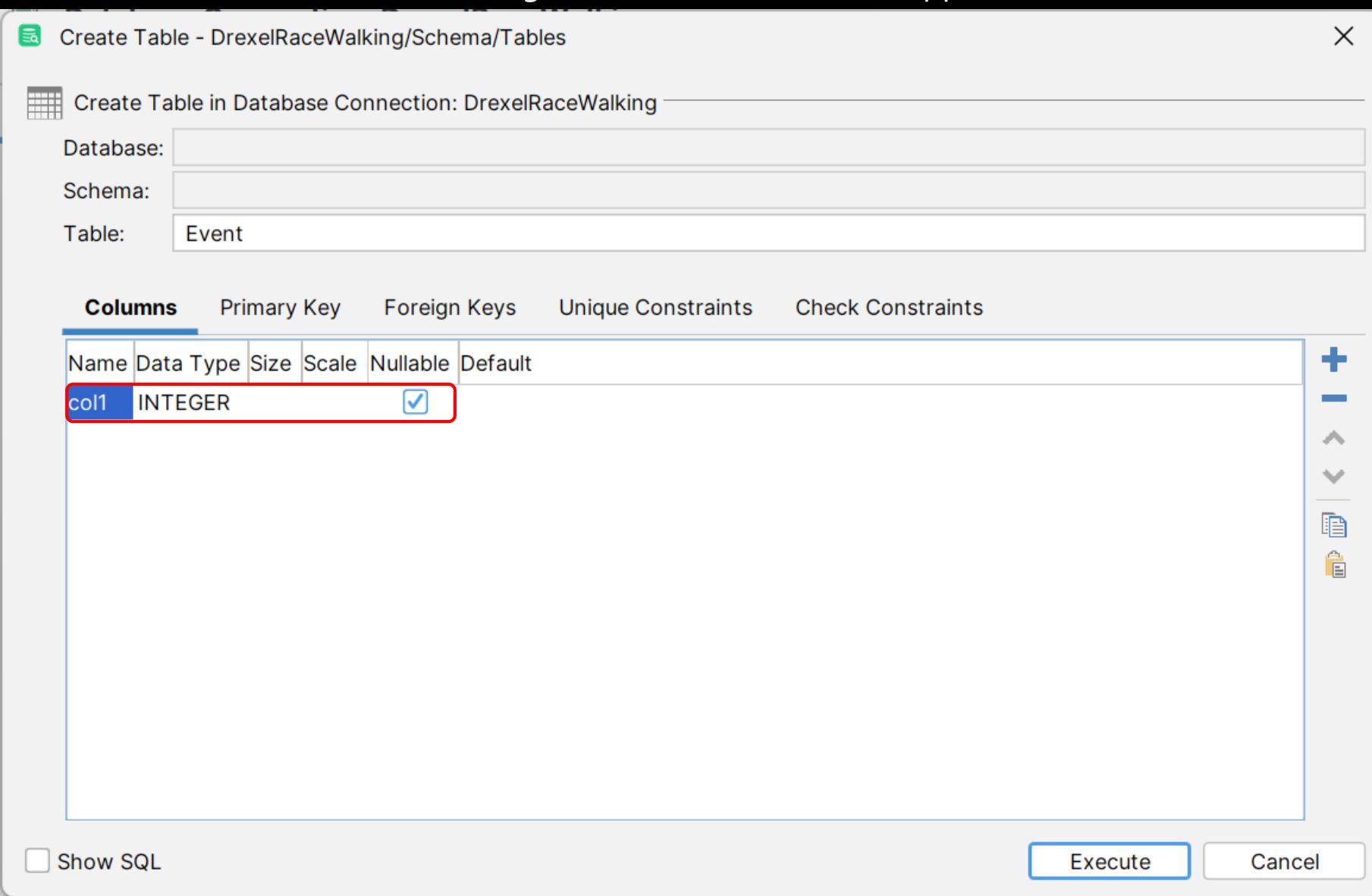
Table: Event

**Columns** Primary Key Foreign Keys Unique Constraints Check Constraints

Name	Data Type	Size	Scale	Nullable	Default
col1	INTEGER			<input checked="" type="checkbox"/>	

**+** **-** **▲** **▼** **📄** **📁**

Show SQL Execute Cancel



# Creating Tables - GUI

Add the *IDEvent* field, change the values to *IDEvent*, leave *INTEGER*:

Create Table - DrexelRaceWalking/Schema/Tables

Create Table in Database Connection: DrexelRaceWalking

Database:

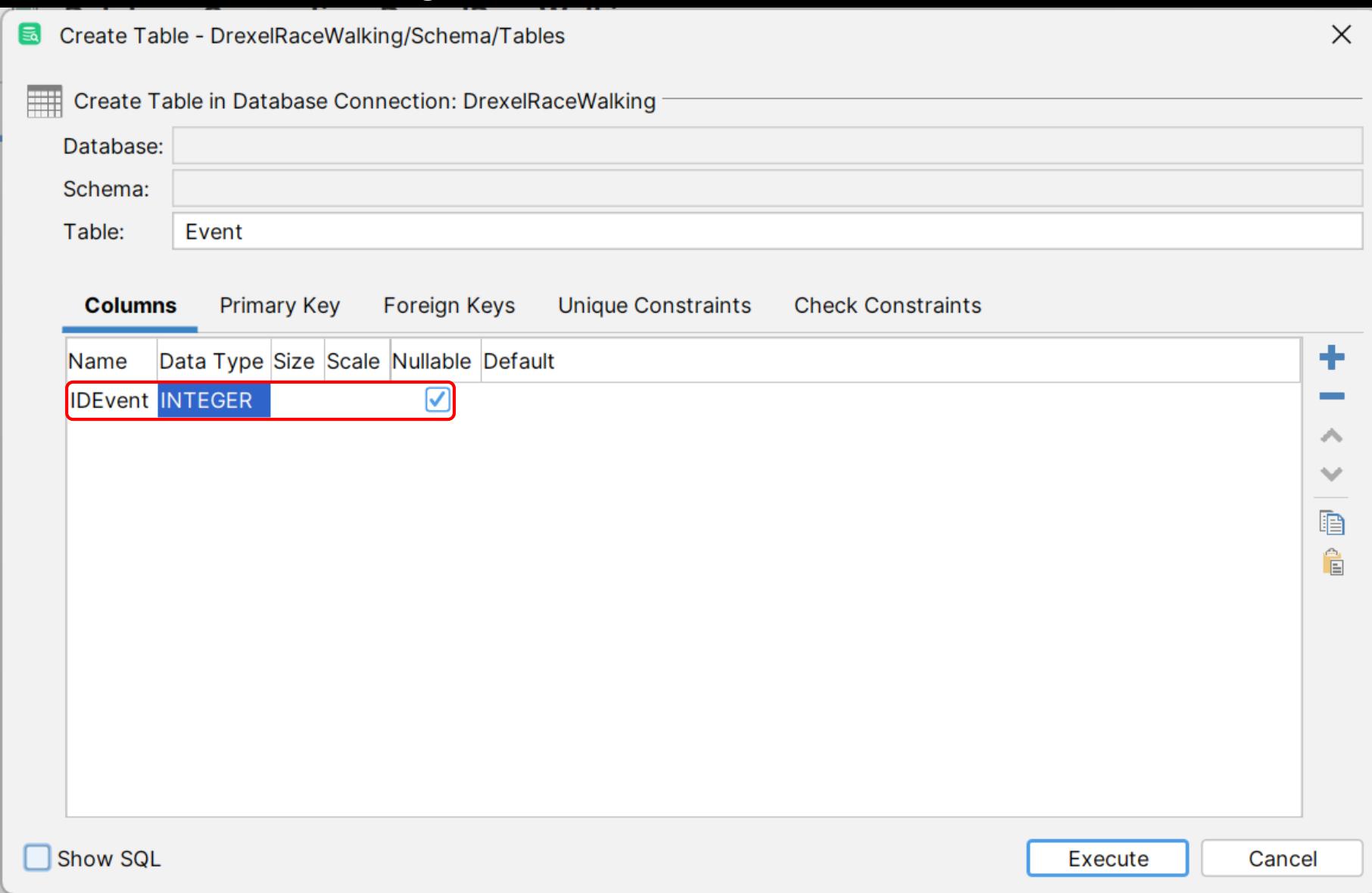
Schema:

Table: Event

**Columns** Primary Key Foreign Keys Unique Constraints Check Constraints

Name	Data Type	Size	Scale	Nullable	Default
IDEvent	INTEGER			<input checked="" type="checkbox"/>	

Show SQL



# Creating Tables - GUI

Uncheck *Nullable*, because *IDEvent* is the primary key.

Create Table - DrexelRaceWalking/Schema/Tables X

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

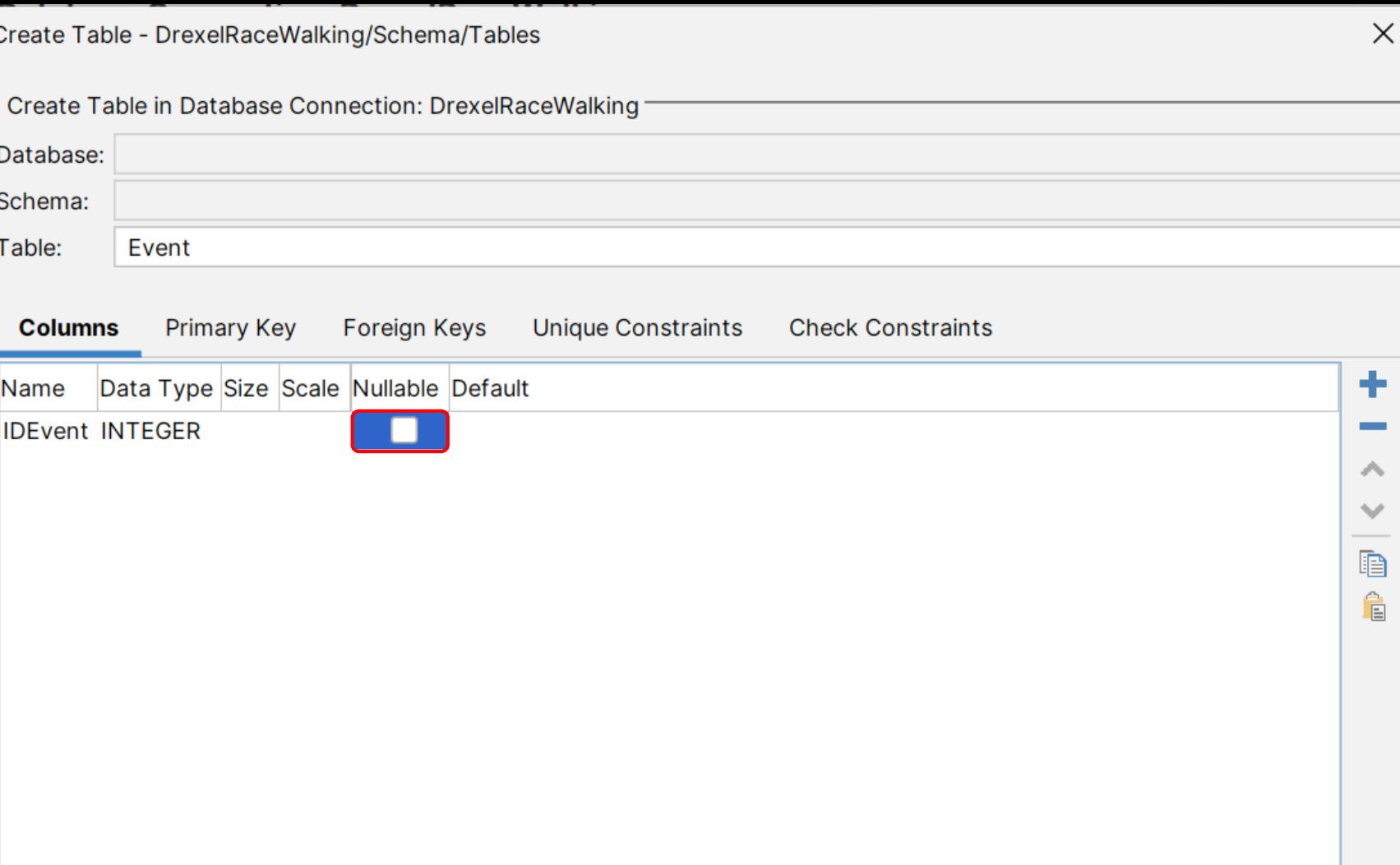
Table: Event

**Columns** Primary Key Foreign Keys Unique Constraints Check Constraints

Name	Data Type	Size	Scale	Nullable	Default
IDEvent	INTEGER			<input checked="" type="checkbox"/>	

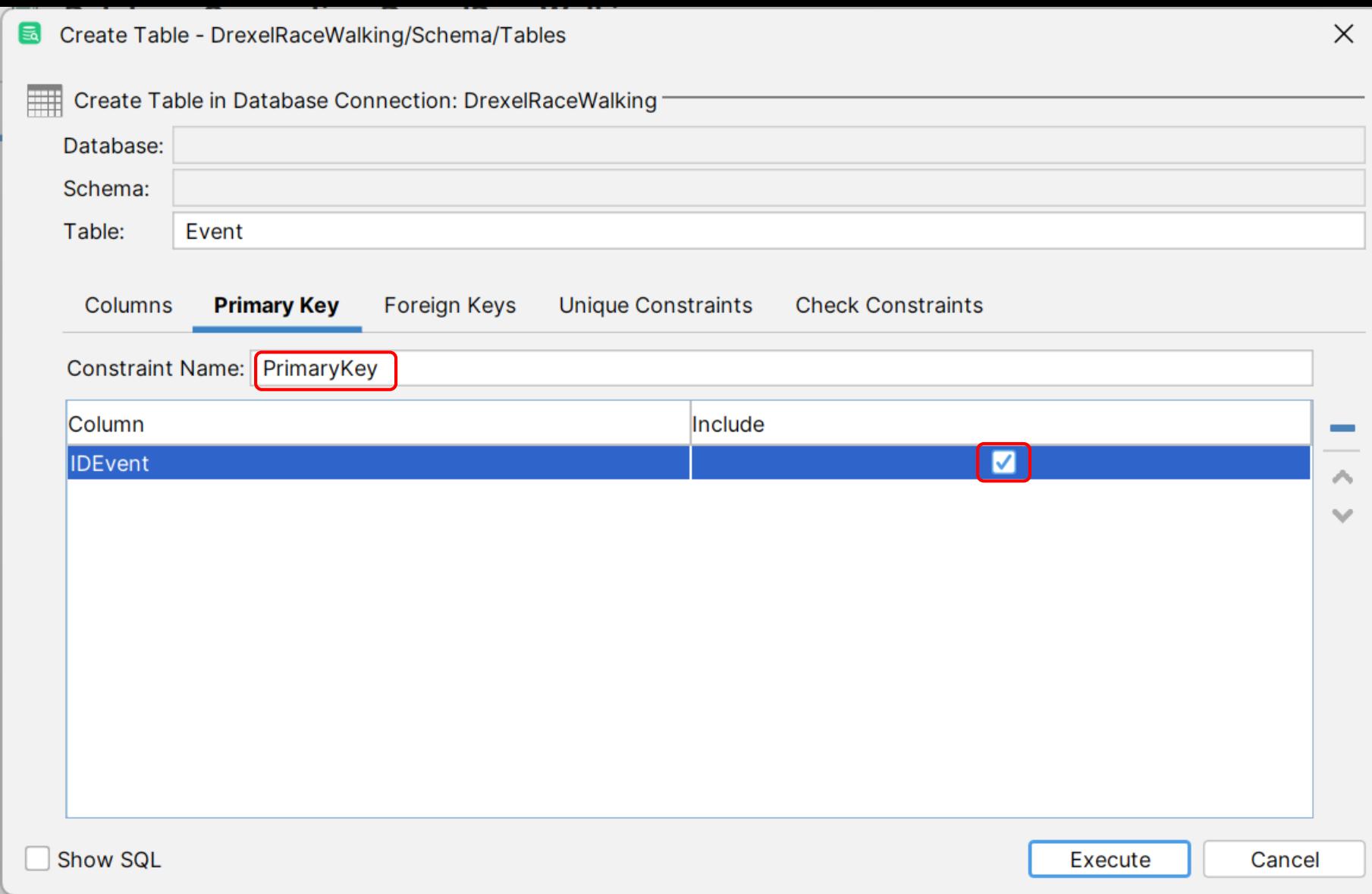
**Actions:** + - ^ v S D

Show SQL Execute Cancel



# Creating Tables - GUI

Click on the *Primary Key* tab, give the *Constraint Name* a value, and check the box to include the *IDEvent* field:



# Creating Tables - GUI

Click on the *Columns* tab so we can enter the rest of the fields:

Create Table - DrexelRaceWalking/Schema/Tables X

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

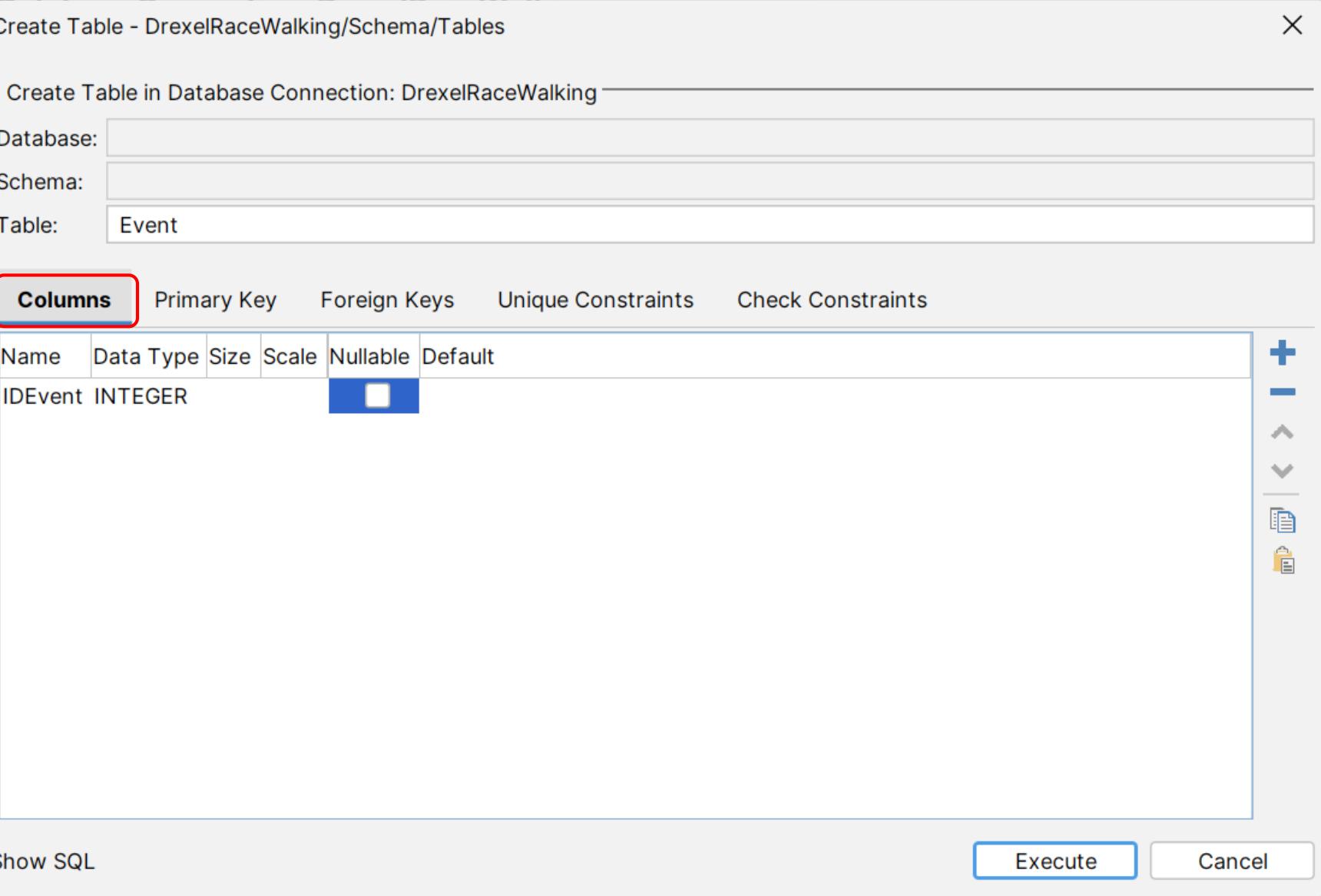
Table: Event

**Columns** Primary Key Foreign Keys Unique Constraints Check Constraints

Name	Data Type	Size	Scale	Nullable	Default
IDEvent	INTEGER			<input checked="" type="checkbox"/>	

+ - ^ v [ ] [ ]

Show SQL Execute Cancel



# Creating Tables - GUI

Enter the remaining fields, all should not be allowed to contain nulls, then click *Execute*:

Create Table - DrexelRaceWalking/Schema/Tables X

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

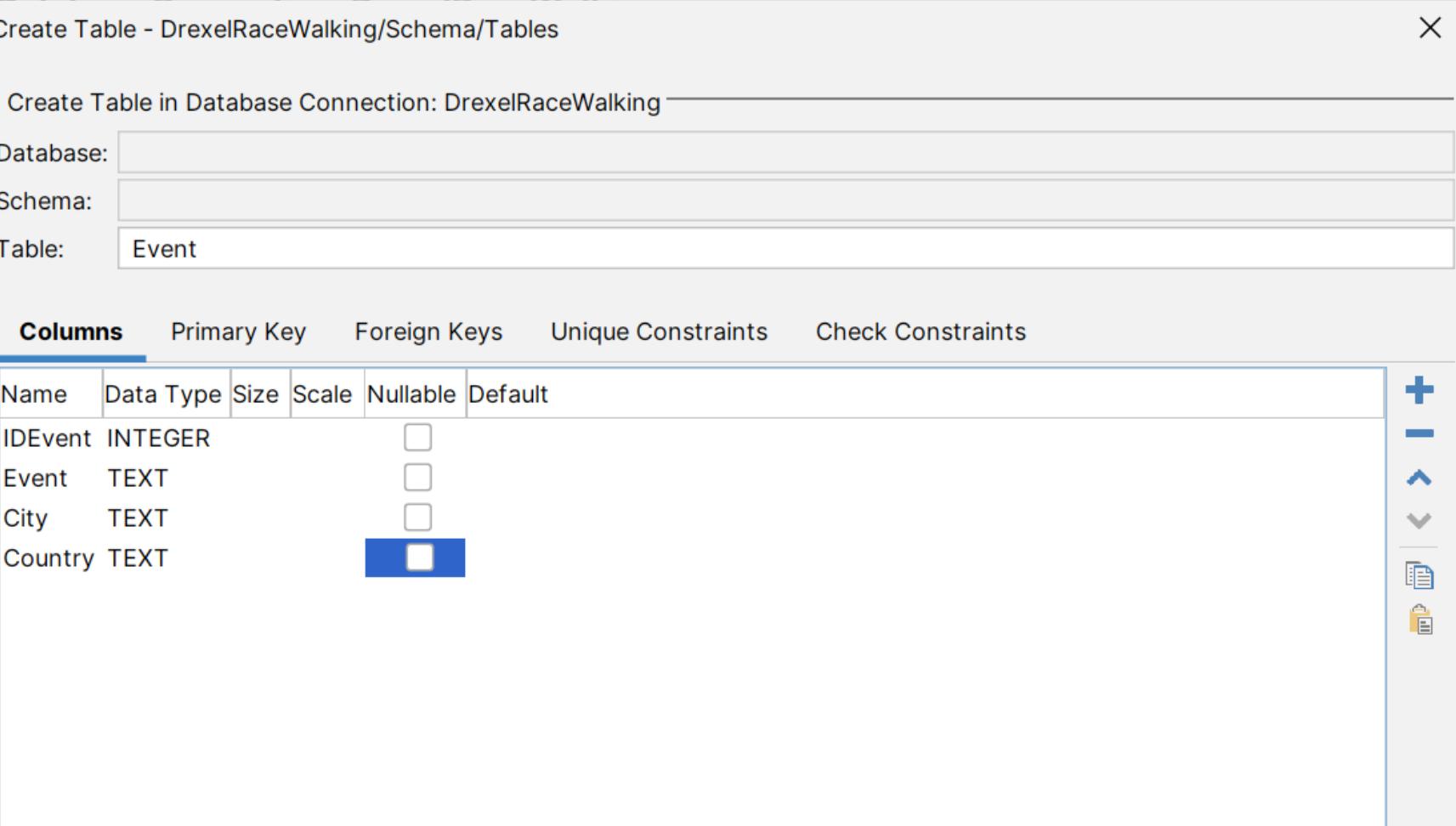
Table: Event

**Columns** Primary Key Foreign Keys Unique Constraints Check Constraints

Name	Data Type	Size	Scale	Nullable	Default
IDEvent	INTEGER			<input type="checkbox"/>	
Event	TEXT			<input type="checkbox"/>	
City	TEXT			<input type="checkbox"/>	
Country	TEXT			<input checked="" type="checkbox"/>	

Collation:

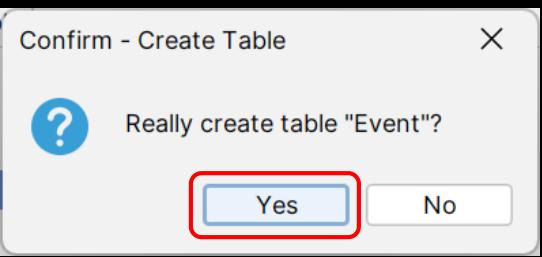
Show SQL



The screenshot shows a 'Create Table' dialog box from a database application. At the top, it says 'Create Table - DrexelRaceWalking/Schema/Tables'. Below that, it says 'Create Table in Database Connection: DrexelRaceWalking'. There are three dropdown menus: 'Database', 'Schema', and 'Table', with 'Event' selected for the Table. The main area is titled 'Columns' and contains a table with four rows. The first row has 'IDEvent' as the name, 'INTEGER' as the data type, and a checkbox for nullable which is unchecked. The second row has 'Event' as the name, 'TEXT' as the data type, and a checkbox for nullable which is unchecked. The third row has 'City' as the name, 'TEXT' as the data type, and a checkbox for nullable which is unchecked. The fourth row has 'Country' as the name, 'TEXT' as the data type, and a checkbox for nullable which is checked. To the right of the table is a vertical toolbar with icons for adding a row, deleting a row, moving rows up and down, and viewing the table structure. At the bottom left is a 'Collation' dropdown with a dropdown arrow. At the bottom right are two buttons: 'Execute' (which is highlighted with a red border) and 'Cancel'.

# Creating Tables - GUI

Click **Yes** to confirm the table creation:



# Creating Tables - GUI

Enter the remaining fields, all should not be allowed to contain nulls, then click *Execute*:

The screenshot shows the DbVisualizer Pro 23.2.5 interface. The title bar reads "DbVisualizer Pro 23.2.5 - DrexelRaceWalking". The left sidebar has tabs for "Databases", "Scripts", and "Favorites". Below these are sections for "Connections" and "DrexelRaceWalking". Under "DrexelRaceWalking", there are "Schema", "Tables", "Views", "Indexes", and "Triggers". The "Tables" item is selected and highlighted with a red box around the "Event" table. The main pane is titled "Database Connection: DrexelRaceWalking" and shows the connection details for "jdbc:sqlite::I:\Drexel\GradDB\DrexelRaceWalking.db". The "Connection" tab is active, displaying the following settings:

Setting	Value
Name	DrexelRaceWalking
Notes	(empty)
Database Type	SQLite
Driver Type	SQLite
Database file name	I:\Drexel\GradDB\DrexelRaceWalking.db
Auto Commit	<input checked="" type="checkbox"/>
Save Database Password	(empty)
Permission Mode	Development

At the bottom of the connection pane, there are buttons for "Reconnect", "Disconnect", and "Ping Server". A "Connection Message" section at the very bottom displays the following text:  
SQLite  
3.41.2  
SQLite JDBC  
3.41.2.1

# Creating Tables - GUI

Quickly spy the columns using the arrows:

File Edit View Database SQL Commander Tools Window Help DbVisualizer Pro 23.2.5 - DrexelRaceWalking

<Drop your favorite objects here>

Databases Scripts Favorites

Connections

DrexelRaceWalking Schema Tables Event Columns IDEvent INTEGER Event TEXT City TEXT Country TEXT Indexes Triggers sqlite\_schema Views Indexes Triggers

SQLite Microsoft Access (UCanAccess)

Database Connection: DrexelRaceWalking  
jdbc:sqlite::|Drexel\GradDB\|DrexelRaceWalking.db Connected - 13:36:11

Connection Properties Database Info Data Types Search Actions...

**Connection**

Name: DrexelRaceWalking  
Notes:

**Database**

Settings Format: Server Info  
Database Type: SQLite  
Driver Type:  SQLite  
Database file name: I:\Drexel\GradDB\|DrexelRaceWalking.db

**Authentication**

Database Userid:   
Database Password:

**Options**

Auto Commit:   
Save Database Password:   
Save Between Sessions:   
Permission Mode: Development

Reconnect Disconnect Ping Server

Connection Message

SQLite 3.41.2  
SQLite JDBC 3.41.2.1

194M of 2048M

# Creating Tables - GUI

Add the *Athlete* table in the same manner (Primary Key is *IDAthlete*):

Create Table - DrexelRaceWalking/Schema/Tables

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: Athlete

**Columns** Primary Key Foreign Keys Unique Constraints Check Constraints

Name	Data Type	Size	Scale	Nullable	Default
IDAthlete	INTEGER			<input type="checkbox"/>	
FirstName	TEXT			<input type="checkbox"/>	
LastName	TEXT			<input type="checkbox"/>	
CountryCode	TEXT			<input checked="" type="checkbox"/>	

**NOTE:** We are missing Gender. A Text field for Gender should be here, non-nullable

Collation:  ▾

Show SQL

Execute Cancel

The screenshot shows a 'Create Table' dialog box. At the top, it says 'Create Table - DrexelRaceWalking/Schema/Tables'. Below that, 'Create Table in Database Connection: DrexelRaceWalking'. Under 'Database' and 'Schema', there are input fields. The 'Table' field contains 'Athlete'. Below this, there are tabs for 'Columns', 'Primary Key', 'Foreign Keys', 'Unique Constraints', and 'Check Constraints'. The 'Columns' tab is selected. A table lists columns: 'Name' (IDAthlete, TEXT, TEXT, TEXT), 'Data Type' (INTEGER, TEXT, TEXT), 'Size' (empty), 'Scale' (empty), 'Nullable' (checkboxes for all three), and 'Default' (empty). The 'CountryCode' row has its 'Nullable' checkbox checked. To the right of the table is a vertical toolbar with icons for adding, deleting, moving, and saving. A note at the bottom left says 'NOTE: We are missing Gender. A Text field for Gender should be here, non-nullable'. At the bottom, there's a 'Collation' dropdown, a 'Show SQL' checkbox, and 'Execute' and 'Cancel' buttons.

# Creating Tables - GUI

Add the *Judge* table in the same manner (Primary Key is *IDJudge*):

Create Table - DrexelRaceWalking/Schema/Tables X

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: Judge

**Columns** Primary Key Foreign Keys Unique Constraints Check Constraints

Name	Data Type	Size	Scale	Nullable	Default
IDJudge	INTEGER			<input type="checkbox"/>	
FirstName	TEXT			<input type="checkbox"/>	
LastName	TEXT			<input type="checkbox"/>	
CountryCode	TEXT			<input checked="" type="checkbox"/>	

Collation:  ▼

Show SQL Execute Cancel

# Creating Tables - GUI

Add the rows for the *Observer* table in the same manner:

Create Table - DrexelRaceWalking/Schema/Tables X

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: Observer

**Columns** Primary Key Foreign Keys Unique Constraints Check Constraints

Name	Data Type	Size	Scale	Nullable	Default
IDObserver	INTEGER			<input checked="" type="checkbox"/>	
FirstName	TEXT			<input type="checkbox"/>	
LastName	TEXT			<input type="checkbox"/>	

+ — ▲ ▼ 📄 📁

Show SQL Execute Cancel

# Creating Tables - GUI

Add the Primary Key for the *Observer* table (*IDObserver*):

Create Table - DrexelRaceWalking/Schema/Tables X

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: Observer

Columns Primary Key Foreign Keys Unique Constraints Check Constraints

Constraint Name: PrimaryKey

Column	Include
IDObserver	<input checked="" type="checkbox"/>
FirstName	<input type="checkbox"/>
LastName	<input type="checkbox"/>

Show SQL Execute Cancel

# Creating Tables - GUI

The tables:

- Event
- Athlete
- Judge
- Observer

All had one thing in common:

There were no fields defined / related in other tables.

They were entity defining tables.

# Creating Tables - GUI

The next table, *Race* is a little different. Start creating the basic fields, all not nullable.

Create Table - DrexelRaceWalking/Schema/Tables X

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: Race

**Columns** Primary Key Foreign Keys Unique Constraints Check Constraints

Name	Data Type	Size	Scale	Nullable	Default
IDRace	INTEGER			<input type="checkbox"/>	
IDEvent	INTEGER			<input type="checkbox"/>	
RaceDate	TEXT			<input type="checkbox"/>	
StartTime	TEXT			<input type="checkbox"/>	
Distance	INTEGER			<input type="checkbox"/>	
DistanceUnits	TEXT			<input type="checkbox"/>	
Gender	TEXT			<input checked="" type="checkbox"/>	

Collation:    
  
  


Show SQL Execute Cancel

# Creating Tables - GUI

The Primary Key is *IDRace*. Note the original video showed this as a compound key including IDEvent. That was a typo

Create Table - DrexelRaceWalking/Schema/Tables

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: Race

Columns Primary Key Foreign Keys Unique Constraints Check Constraints

Constraint Name: PrimaryKey

Column	Include
IDRace	<input checked="" type="checkbox"/>
IDEVENT	<input type="checkbox"/>
RaceDate	<input type="checkbox"/>
StartTime	<input type="checkbox"/>
Distance	<input type="checkbox"/>
DistanceUnits	<input type="checkbox"/>
Gender	<input type="checkbox"/>

Show SQL

Execute Cancel

# Creating Tables - GUI

We are not done. We need to ensure that *IDRace* exists. Click on the *Foreign Keys* tab:

Create Table - DrexelRaceWalking/Schema/Tables X

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: Race

Columns Primary Key Foreign Keys Unique Constraints Check Constraints

Foreign Keys tab selected (highlighted with red box)

Constraints

Constraint Name	Columns	On Delete Action	On Update Action

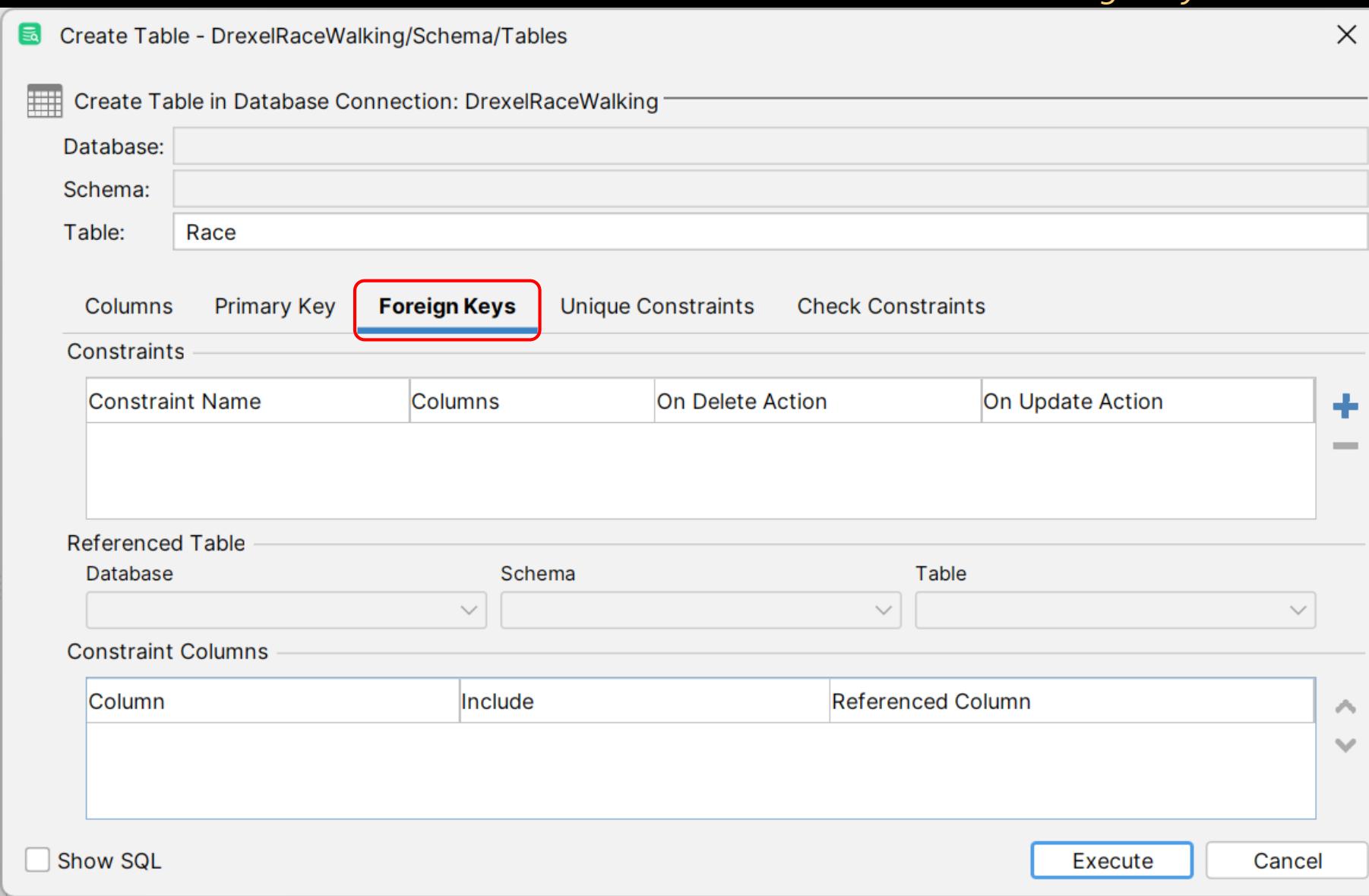
Referenced Table

Database Schema Table

Constraint Columns

Column	Include	Referenced Column

Show SQL  Execute Cancel



# Creating Tables - GUI

We are not done. We need to ensure that *IEvent* exists. Click on the *Foreign Keys* tab:

Create Table - DrexelRaceWalking/Schema/Tables X

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: Race

Columns Primary Key Foreign Keys Unique Constraints Check Constraints

Constraints

Constraint Name	Columns	On Delete Action	On Update Action
			<span style="border: 2px solid red; padding: 2px;">+</span>

Referenced Table

Database  Schema  Table

Constraint Columns

Column	Include	Referenced Column

Show SQL Execute Cancel

# Creating Tables - GUI

Select the lookup table, Event. Include the *IDEvent* column. Select the *IDEvent* reference column. Hit the *Execute* button:

Create Table - DrexelRaceWalking/Schema/Tables X

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: Race

Columns Primary Key Foreign Keys Unique Constraints Check Constraints

Foreign Keys tab selected.

Constraints

Constraint Name	Columns	On Delete Action	On Update Action
Race_fk1	IDEvent		

+   
 -

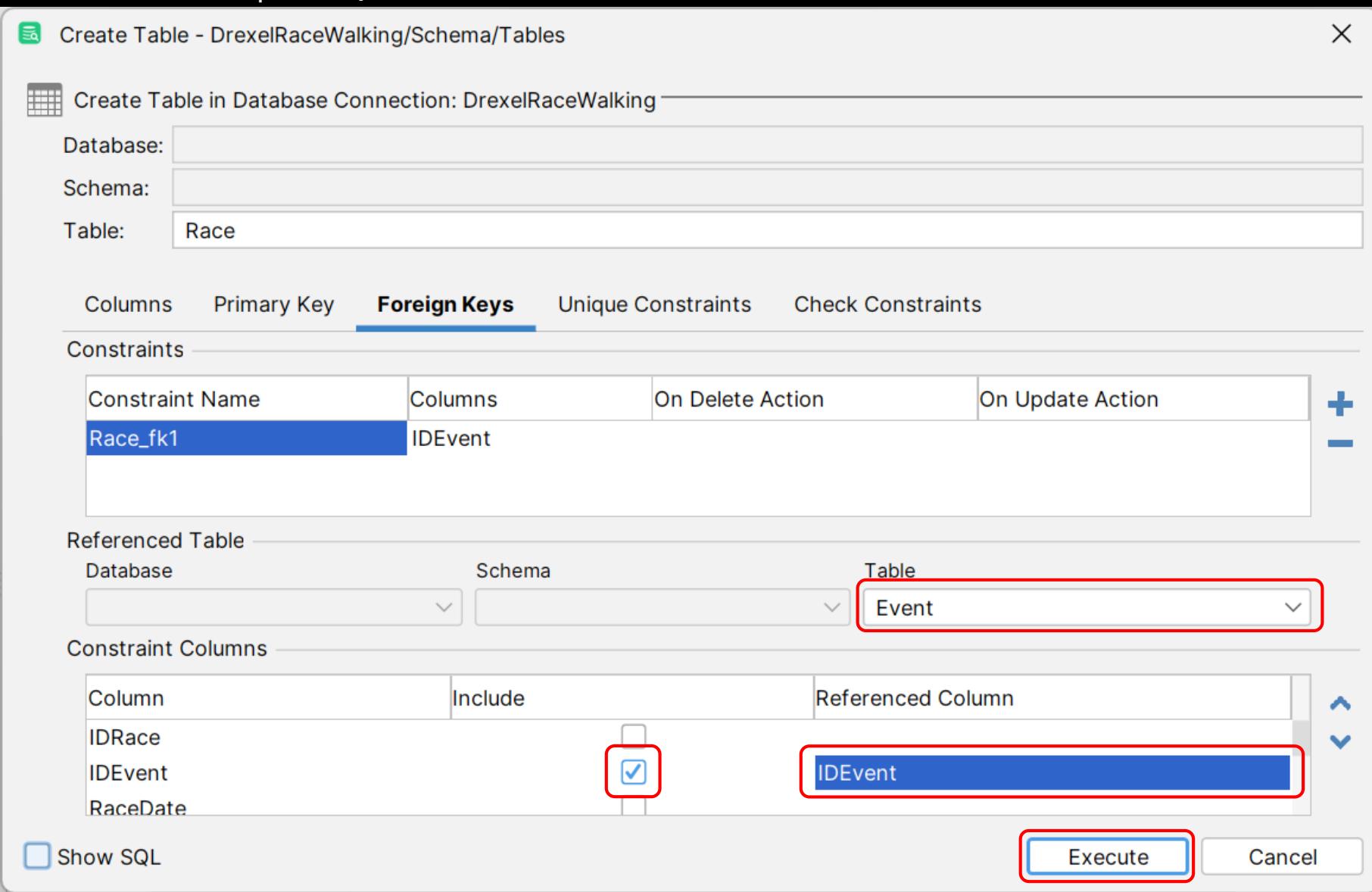
Referenced Table

Database  Schema  Table

Constraint Columns

Column	Include	Referenced Column
IDRace	<input checked="" type="checkbox"/>	
IDEvent	<input checked="" type="checkbox"/>	<input style="border: 2px solid red; width: 150px; height: 30px; margin-left: 10px;" type="text" value="IDEvent"/>
RaceDate		

Show SQL  Execute Execute Cancel



# Creating Tables - GUI

Create the *Bib* table. Add the fields:

Create Table - DrexelRaceWalking/Schema/Tables

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: Bib

Columns	Primary Key	Foreign Keys	Unique Constraints	Check Constraints	
Name	Data Type	Size	Scale	Nullable	Default
IDRace	INTEGER			<input type="checkbox"/>	
IDAthlete	INTEGER			<input type="checkbox"/>	
BibNumber	INTEGER			<input type="checkbox"/>	
FinishingTime	TEXT			<input checked="" type="checkbox"/>	
FinishingPlace	INTEGER			<input checked="" type="checkbox"/>	
Finished	TEXT			<input type="checkbox"/>	

+   
 -   
 ▲   
 ▼   
 ↻   
 ↺

Collation:  ▾

Show SQL

Execute Cancel

# Creating Tables - GUI

Add a compound primary key with *IDRace* & *IDAthlete*:

Create Table - DrexelRaceWalking/Schema/Tables X

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: Bib

Columns Primary Key Foreign Keys Unique Constraints Check Constraints

Constraint Name: PrimaryKey

Column	Include
IDRace	<input checked="" type="checkbox"/>
IDAthlete	<input checked="" type="checkbox"/>
BibNumber	<input type="checkbox"/>
FinishingTime	<input type="checkbox"/>
FinishingPlace	<input type="checkbox"/>
Finished	<input type="checkbox"/>

Show SQL  Execute Cancel

# Creating Tables - GUI

Add a foreign key for *IDRace* to the *Race* table:

Create Table - DrexelRaceWalking/Schema/Tables

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: Bib

Columns Primary Key Foreign Keys Unique Constraints Check Constraints

Foreign Keys

Constraints

Constraint Name	Columns	On Delete Action	On Update Action
Bib_fk1	IDRace		

+ -

Referenced Table

Database  Schema  Table

Constraint Columns

Column	Include	Referenced Column
IDRace	<input checked="" type="checkbox"/>	IDRace
IDAthlete	<input type="checkbox"/>	
BibNumber	<input type="checkbox"/>	

Show SQL

Execute Cancel

# Creating Tables - GUI

Add a foreign key for *IDAthlete* to the *Athlete* table:

Create Table - DrexelRaceWalking/Schema/Tables

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: Bib

Columns Primary Key Foreign Keys Unique Constraints Check Constraints

Foreign Keys tab selected.

Constraints

Constraint Name	Columns	On Delete Action	On Update Action
Bib_fk1	IDRace		
Bib_fk2	IDAthlete		

+   
 -

Referenced Table

Database  Schema  Table

Constraint Columns

Column	Include	Referenced Column
IDRace	<input type="checkbox"/>	
IDAthlete	<input checked="" type="checkbox"/>	IDAthlete
BibNumber	<input type="checkbox"/>	

Show SQL

Execute Cancel

# Creating Tables - GUI

Create the *RaceJudge* table:

Create Table - DrexelRaceWalking/Schema/Tables X

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table:

**Columns** Primary Key Foreign Keys Unique Constraints Check Constraints

Name	Data Type	Size	Scale	Nullable	Default
IDRace	INTEGER			<input type="checkbox"/>	
IDJudge	INTEGER			<input checked="" type="checkbox"/>	

+ — ^ v [ ] { }

Show SQL Execute Cancel

# Creating Tables - GUI

Add a compound Primary Key:

Create Table - DrexelRaceWalking/Schema/Tables X

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: RaceJudge

Columns Primary Key Foreign Keys Unique Constraints Check Constraints

Constraint Name: PrimaryKey

Column	Include
IDRace	<input checked="" type="checkbox"/>
IDJudge	<input checked="" type="checkbox"/>

Show SQL  Execute Cancel

# Creating Tables - GUI

Add a Foreign Key for *IDRace*:

Create Table - DrexelRaceWalking/Schema/Tables X

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: RaceJudge

Columns Primary Key **Foreign Keys** Unique Constraints Check Constraints

Constraints

Constraint Name	Columns	On Delete Action	On Update Action
RaceJudge_fk1	IDRace		

+ -

Referenced Table

Database  Schema  Table

Constraint Columns

Column	Include	Referenced Column
IDRace	<input checked="" type="checkbox"/>	IDRace
IDJudge	<input type="checkbox"/>	

Show SQL Execute Cancel

# Creating Tables - GUI

Add a Foreign Key for *IDJudge*:

Create Table - DrexelRaceWalking/Schema/Tables X

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: RaceJudge

Columns Primary Key **Foreign Keys** Unique Constraints Check Constraints

Constraints

Constraint Name	Columns	On Delete Action	On Update Action
RaceJudge_fk1	IDRace		
RaceJudge_fk2	IDJudge		

+ -

Referenced Table

Database  Schema  Table

Constraint Columns

Column	Include	Referenced Column
IDRace	<input type="checkbox"/>	
IDJudge	<input checked="" type="checkbox"/>	IDJudge

▲ ▼

Show SQL Execute Cancel

# Creating Tables - GUI

Create the fields of the *JudgeCall* table:

Create Table - DrexelRaceWalking/Schema/Tables

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: JudgeCall

**Columns** Primary Key Foreign Keys Unique Constraints Check Constraints

Name	Data Type	Size	Scale	Nullable	Default
IDRace	INTEGER			<input type="checkbox"/>	
IDJudge	INTEGER			<input type="checkbox"/>	
Color	TEXT			<input type="checkbox"/>	
Infraction	TEXT			<input type="checkbox"/>	
TOD	TEXT			<input type="checkbox"/>	
BibNumber	INTEGER			<input checked="" type="checkbox"/>	

**+** **-** **▲** **▼** **File** **Help**

Show SQL  Execute Cancel

# Creating Tables - GUI

Add a compound Primary Key on *IDRace*, *IDJudge*, *BibNumber*, *Color*, & *Infraction*:

Create Table - DrexelRaceWalking/Schema/Tables

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: JudgeCall

Columns Primary Key Foreign Keys Unique Constraints Check Constraints

Constraint Name: PrimaryKey

Column	Include
IDRace	<input checked="" type="checkbox"/>
IDJudge	<input checked="" type="checkbox"/>
Color	<input checked="" type="checkbox"/>
Infraction	<input checked="" type="checkbox"/>
TOD	<input type="checkbox"/>
BibNumber	<input checked="" type="checkbox"/>

Show SQL

Execute Cancel

# Creating Tables - GUI

Create a Foreign Key for *IDRace* & *BibNumber* from the *BibNumber* table

Create Table - DrexelRaceWalking/Schema/Tables

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: JudgeCall

Columns Primary Key Foreign Keys Unique Constraints Check Constraints

Foreign Keys tab selected.

Constraints

Constraint Name	Columns	On Delete Action	On Update Action
JudgeCall_fk1	IDRace, BibNumber		

+   
 -

Referenced Table

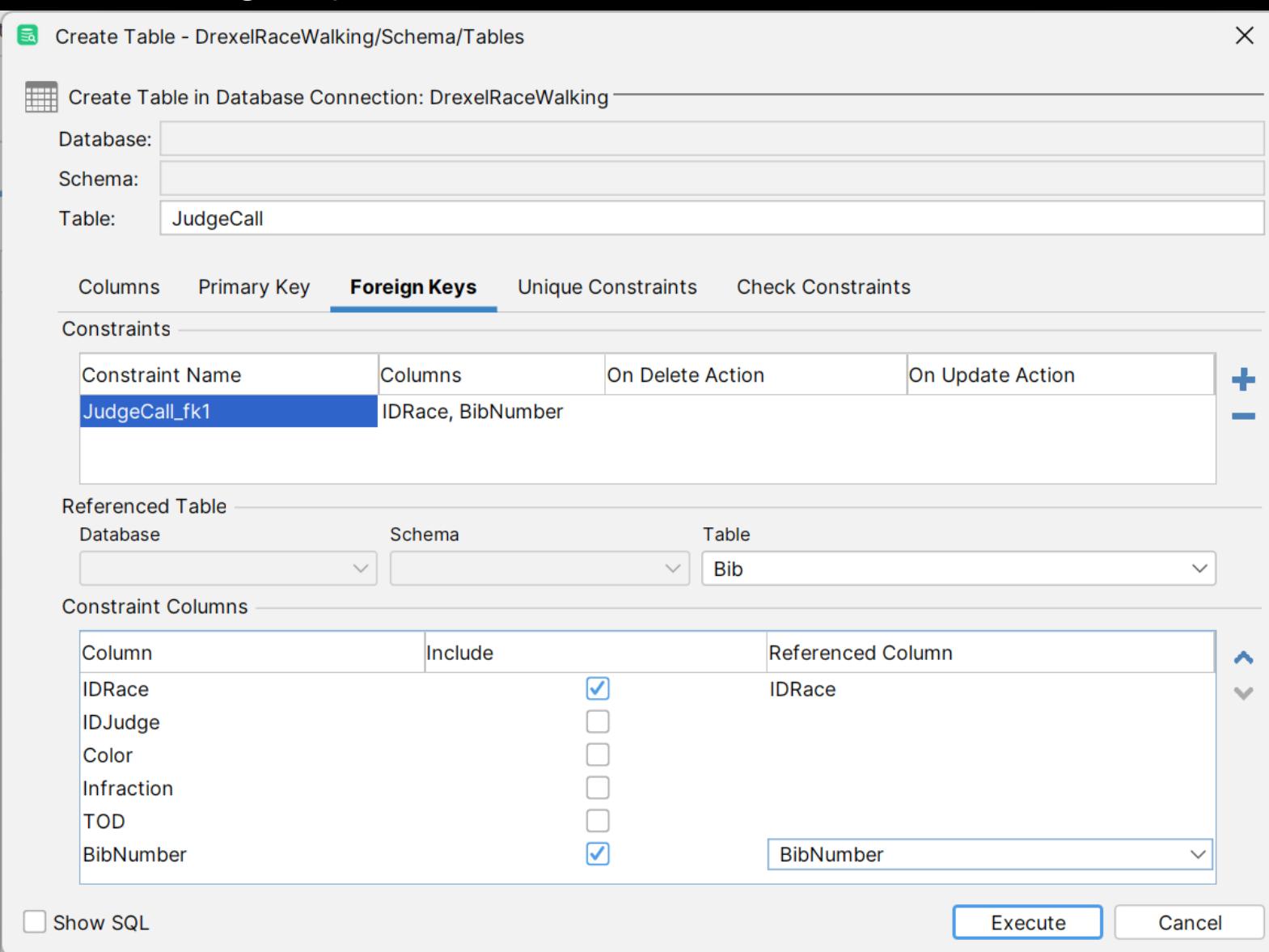
Database:  Schema:  Table: Bib

Constraint Columns

Column	Include	Referenced Column
IDRace	<input checked="" type="checkbox"/>	IDRace
IDJudge	<input type="checkbox"/>	
Color	<input type="checkbox"/>	
Infraction	<input type="checkbox"/>	
TOD	<input type="checkbox"/>	
BibNumber	<input checked="" type="checkbox"/>	BibNumber

Show SQL

Execute Cancel



We can't simply check if the bib number exists, because bib numbers are reused from race to race.

# Creating Tables - GUI

Create a Foreign Key for *IDJudge*

Create Table - DrexelRaceWalking/Schema/Tables

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: JudgeCall

Columns Primary Key Foreign Keys Unique Constraints Check Constraints

Foreign Keys tab selected.

Constraints

Constraint Name	Columns	On Delete Action	On Update Action
JudgeCall_fk1	IDRace, BibNumber		
JudgeCall_fk2	IDJudge		

+   
 -

Referenced Table

Database:  Schema:  Table: Judge

Constraint Columns

Column	Include	Referenced Column
IDRace	<input type="checkbox"/>	
IDJudge	<input checked="" type="checkbox"/>	IDJudge
Color	<input type="checkbox"/>	
Infraction	<input type="checkbox"/>	
TOD	<input type="checkbox"/>	
BibNumber	<input type="checkbox"/>	

Show SQL

Execute Cancel

# Creating Tables - GUI

Create the rows for the *VideoObservation* table:

Create Table - DrexelRaceWalking/Schema/Tables

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: **VideoObservation**

**Columns** Primary Key Foreign Keys Unique Constraints Check Constraints

Name	Data Type	Size	Scale	Nullable	Default
ID	INTEGER			<input checked="" type="checkbox"/>	
IDRace	INTEGER			<input checked="" type="checkbox"/>	
BibNumber	INTEGER			<input checked="" type="checkbox"/>	
IDObserver	INTEGER			<input checked="" type="checkbox"/>	
LOCAverage	NUMERIC			<input checked="" type="checkbox"/>	
LOC1	NUMERIC			<input checked="" type="checkbox"/>	
LOC2	NUMERIC			<input checked="" type="checkbox"/>	
LOC3	NONE			<input checked="" type="checkbox"/>	
LOC4	NUMERIC			<input checked="" type="checkbox"/>	
KneeAngle	NUMERIC			<input checked="" type="checkbox"/>	
Comment	TEXT			<input checked="" type="checkbox"/>	
VideoFile	TEXT			<input checked="" type="checkbox"/>	
<b>TOD</b>	TEXT			<input checked="" type="checkbox"/>	

Collation:

Show SQL

# Creating Tables - GUI

Create the Primary Key for the *VideoObservation* table:

Create Table - DrexelRaceWalking/Schema/Tables

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: VideoObservation

Columns Primary Key Foreign Keys Unique Constraints Check Constraints

Constraint Name: PrimaryKey

Column	Include
ID	<input checked="" type="checkbox"/>
IDRace	<input type="checkbox"/>
BibNumber	<input type="checkbox"/>
IDObserver	<input type="checkbox"/>
LOCAverage	<input type="checkbox"/>
LOC1	<input type="checkbox"/>
LOC2	<input type="checkbox"/>
LOC3	<input type="checkbox"/>
LOC4	<input type="checkbox"/>
KneeAngle	<input type="checkbox"/>
Comment	<input type="checkbox"/>
VideoFile	<input type="checkbox"/>
TOD	<input type="checkbox"/>

Show SQL

Execute Cancel

The choice of Primary Key could be debated.

Do we need an *ID* field?

We could say that *IDRace*, *BibNumber*, *VideoFile*, & *TOD* make a unique record.  
Sometimes the choice is style.

# Creating Tables - GUI

Create a Foreign Key for the *IDRace* and *BibNumber* fields to the *Bib* table:

Create Table - DrexelRaceWalking/Schema/Tables

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: VideoObservation

Columns Primary Key Foreign Keys Unique Constraints Check Constraints

Constraints

Constraint Name	Columns	On Delete Action	On Update Action
VideoObservation_fk1	IDRace, BibNumber		

+  
-

Referenced Table

Database  Schema  Table

Constraint Columns

Column	Include	Referenced Column
ID	<input type="checkbox"/>	
IDRace	<input checked="" type="checkbox"/>	IDRace
BibNumber	<input checked="" type="checkbox"/>	BibNumber
IDObserver	<input type="checkbox"/>	
LOCAverage	<input type="checkbox"/>	
LOC1	<input type="checkbox"/>	

Show SQL

Execute Cancel

# Creating Tables - GUI

Create a Foreign Key for the *IDObserver* field to the *Observer* table:

Create Table - DrexelRaceWalking/Schema/Tables

Create Table in Database Connection: DrexelRaceWalking

Database:

Schema:

Table: VideoObservation

Columns Primary Key Foreign Keys Unique Constraints Check Constraints

Constraints

Constraint Name	Columns	On Delete Action	On Update Action
VideoObservation_fk1	IDRace, BibNumber		
VideoObservation_fk2	IDObserver		

+    -

Referenced Table

Database  Schema  Table

Constraint Columns

Column	Include	Referenced Column
ID	<input type="checkbox"/>	
IDRace	<input type="checkbox"/>	
BibNumber	<input type="checkbox"/>	
IDObserver	<input checked="" type="checkbox"/>	IDObserver
LOCAverage	<input type="checkbox"/>	
LOC1	<input type="checkbox"/>	

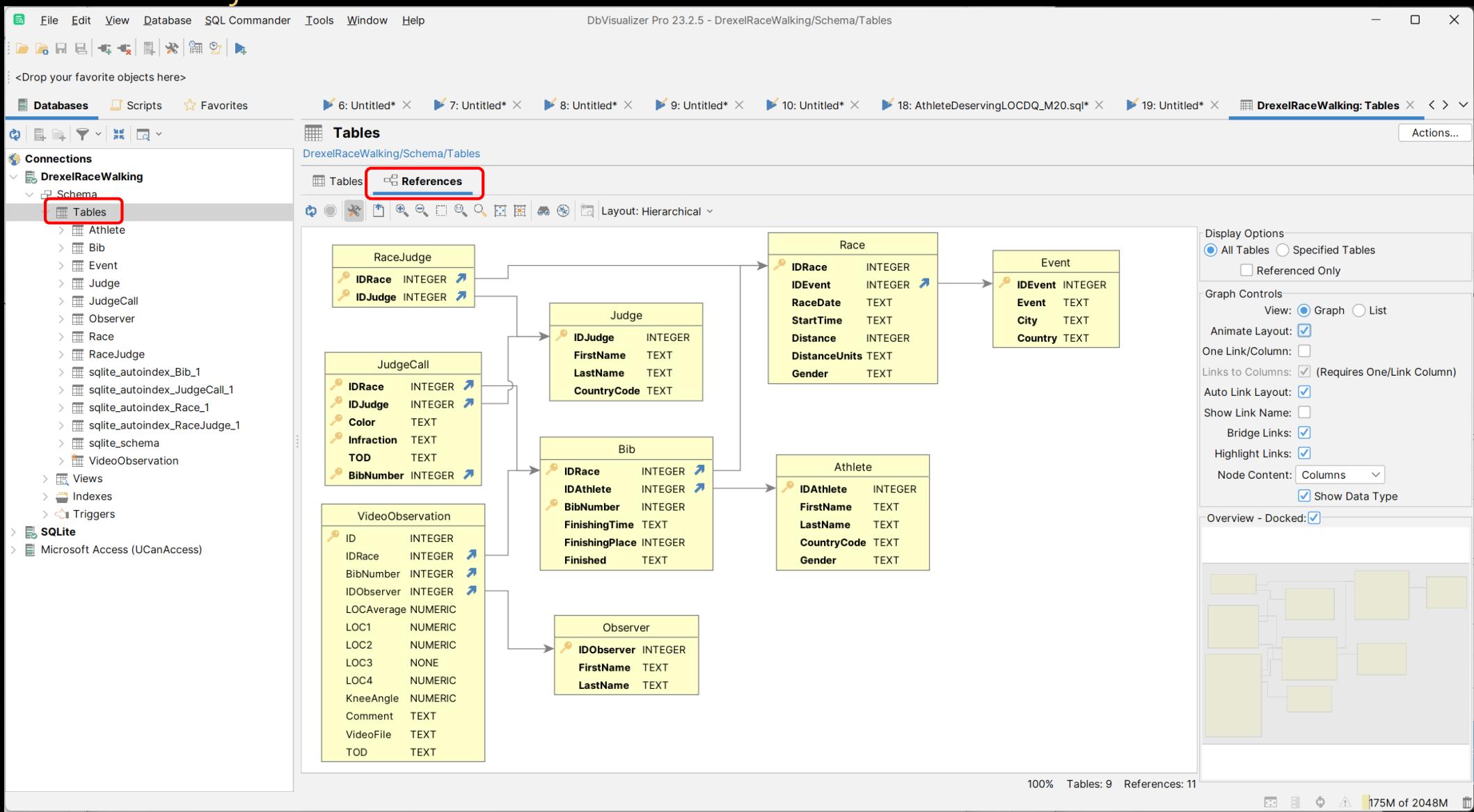
Show SQL

Execute Cancel

# Visualizing the Database

Let's look at what we created.

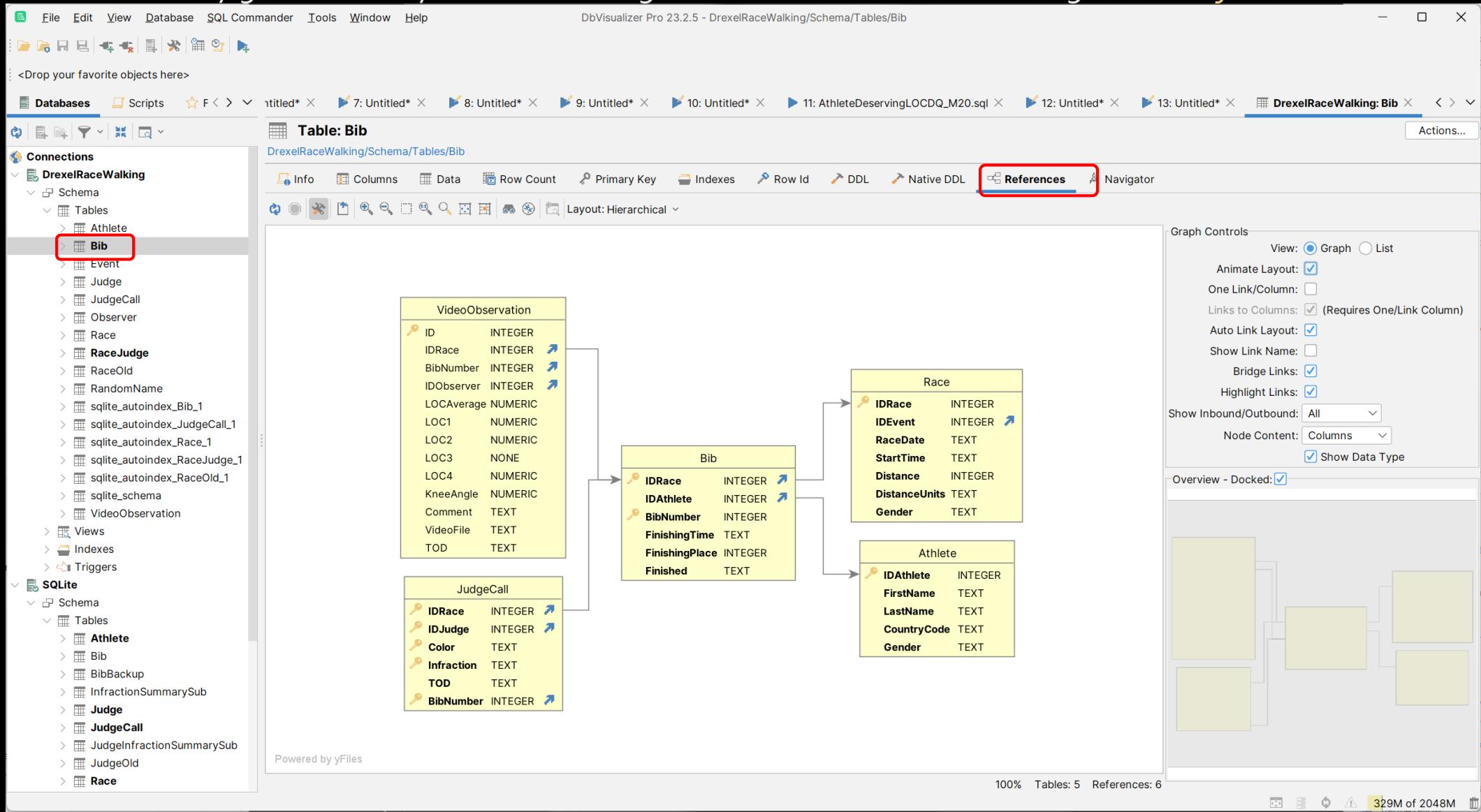
Double-click on *Tables* then click on *References*:



# Visualizing the Database

Let's look at what we created.

We can also see the references for any given table by Double-clicking on the table name and clicking on the *References* tab:



# Creating Tables - Code

Create the *Event* table in code:

```
CREATE TABLE
    Event
    (
        IDEvent INTEGER NOT NULL,
        Event TEXT NOT NULL,
        City TEXT NOT NULL,
        Country TEXT NOT NULL,
        CONSTRAINT PrimaryKey PRIMARY KEY (IDEVENT)
    )
```

# Creating Tables - Code

Create the *Athlete* table in code:

```
CREATE TABLE
    AthleteTest
(
    IDAthlete    INTEGER NOT NULL,
    FirstName    TEXT NOT NULL,
    LastName     TEXT NOT NULL,
    CountryCode  TEXT NOT NULL,
    Gender       TEXT NOT NULL,
    CONSTRAINT PrimaryKey PRIMARY KEY (IDAthlete)
)
```

# Creating Tables - Code

Create the *Judge* table in code:

```
CREATE TABLE
    Judge
(
    IDJudge      INTEGER NOT NULL,
    FirstName    TEXT NOT NULL,
    LastName     TEXT NOT NULL,
    CountryCode  TEXT NOT NULL,
    CONSTRAINT PrimaryKey PRIMARY KEY (IDJudge)
)
```

# Creating Tables - Code

Create the *Race* table in code:

```
CREATE TABLE
    Race
    (
        IDRace      INTEGER NOT NULL,
        IDEvent     INTEGER NOT NULL,
        RaceDate    TEXT NOT NULL,
        StartTime   TEXT NOT NULL,
        Distance    INTEGER NOT NULL,
        DistanceUnits TEXT NOT NULL,
        Gender      TEXT NOT NULL,
        CONSTRAINT PrimaryKey PRIMARY KEY (IDRace),
        CONSTRAINT Race_fk1 FOREIGN KEY (IDEVENT) REFERENCES "Event" ("IDEVENT")
    )
```

# Creating Tables - Code

Create the *Bib* table in code:

```
CREATE TABLE  
    Bib  
(  
    IDRace      INTEGER NOT NULL,  
    IDAthlete   INTEGER NOT NULL,  
    BibNumber   INTEGER NOT NULL,  
    FinishingTime TEXT,  
    FinishingPlace INTEGER,  
    Finished     TEXT NOT NULL,  
    CONSTRAINT PrimaryKey PRIMARY KEY (IDRace, BibNumber),  
    CONSTRAINT Bib_fk1 FOREIGN KEY (IDRace) REFERENCES "Race" ("IDRace"),  
    CONSTRAINT Bib_fk2 FOREIGN KEY (IDAthlete) REFERENCES "Athlete" ("IDAthlete")  
)
```

PLEASE NOTE, Prof. Salvage updated this slide to reflect FinishingTime & FinishingPlace allowing NULL.

# Creating Tables - Code

Create the *RaceJudge* table in code:

```
CREATE TABLE
    RaceJudge
(
    IDRace  INTEGER NOT NULL,
    IDJudge INTEGER NOT NULL,
    CONSTRAINT PrimaryKey PRIMARY KEY (IDRace, IDJudge),
    CONSTRAINT RaceJudge_fk1 FOREIGN KEY (IDRace) REFERENCES "Race" ("IDRace"),
    CONSTRAINT RaceJudge_fk2 FOREIGN KEY (IDJudge) REFERENCES "Judge" ("IDJudge")
)
```

# Creating Tables - Code

Create the *JudgeCall* table in code:

```
CREATE TABLE
  JudgeCall
  (
    IDRace      INTEGER NOT NULL,
    IDJudge     INTEGER NOT NULL,
    Color       TEXT NOT NULL,
    Infraction  TEXT NOT NULL,
    TOD         TEXT NOT NULL,
    BibNumber   INTEGER NOT NULL,
    CONSTRAINT PrimaryKey PRIMARY KEY (IDRace, IDJudge, Color, Infraction, BibNumber),
    CONSTRAINT JudgeCall_fk1 FOREIGN KEY (IDRace, BibNumber) REFERENCES "Bib" ("IDRace",
    "BibNumber"),
    CONSTRAINT JudgeCall_fk2 FOREIGN KEY (IDJudge) REFERENCES "Judge" ("IDJudge")
  )
```

# Creating Tables - Code

Create the *Observer* table in code:

```
CREATE TABLE
    Observer
    (
        IDObserver INTEGER,
        FirstName INTEGER,
        LastName   INTEGER,
        CONSTRAINT PrimaryKey PRIMARY KEY (IDObserver)
    )
```

# Creating Tables - Code

Create the *VideoObservation* table in code:

```
CREATE TABLE
    VideoObservation
(
    ID          INTEGER,
    IDRace      INTEGER,
    BibNumber   INTEGER,
    IDObserver  INTEGER,
    LOCAverage  NUMERIC,
    LOC1        NUMERIC,
    LOC2        NUMERIC,
    LOC3        NUMERIC,
    LOC4        NUMERIC,
    KneeAngle   NUMERIC,
    Comment     TEXT,
    VideoFile   TEXT,
    TOD         TEXT,
    CONSTRAINT PrimaryKey PRIMARY KEY (ID),
    CONSTRAINT VideoObservation_fk1 FOREIGN KEY (IDRace, BibNumber) REFERENCES "Bib" ("IDRace"
    , "BibNumber"),
    CONSTRAINT VideoObservation_fk2 FOREIGN KEY (IDObserver) REFERENCES "Observer"
    ("IDObserver")
)
```

# Creating Tables - Code

Tables can be altered using the `ALTER TABLE` command.

The basic syntax is

```
ALTER TABLE TableName Operation OperationSpecifics;
```

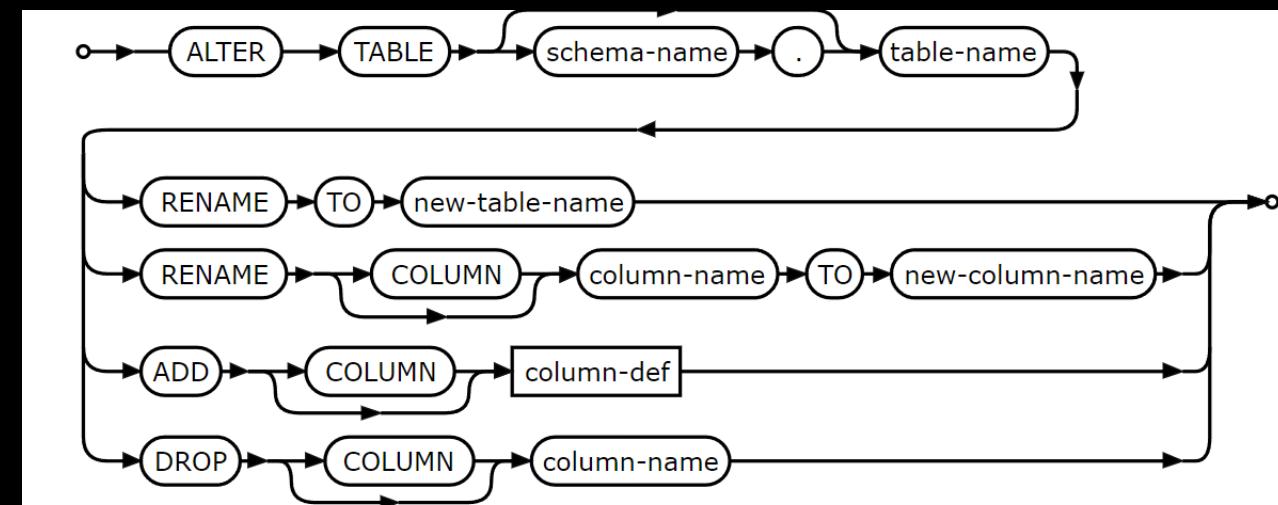
Operations can include:

- Rename
- Add
- Drop

Sqlite doesn't support all expected operations.

Unsupported operations include `DROP COLUMN` in certain situations as well as changing constraints.

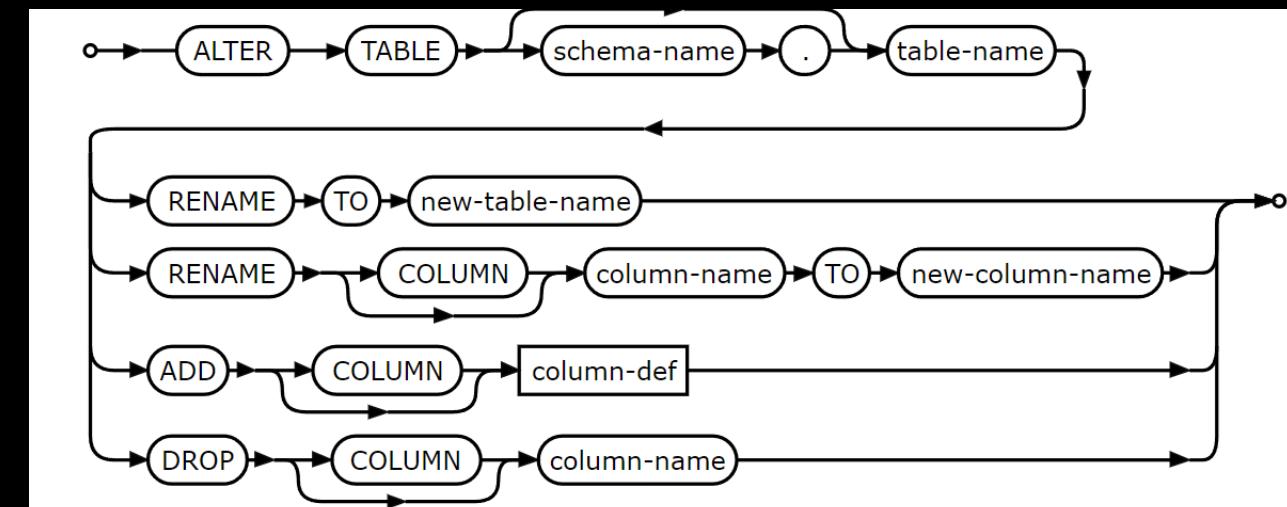
DBVisualizer may be limiting access to changes as well.



# Creating Tables - Code

The following is an example of adding a new column to the *JudgeCall* table called *NewField*:

```
ALTER TABLE JudgeCall ADD COLUMN NewField INTEGER
```

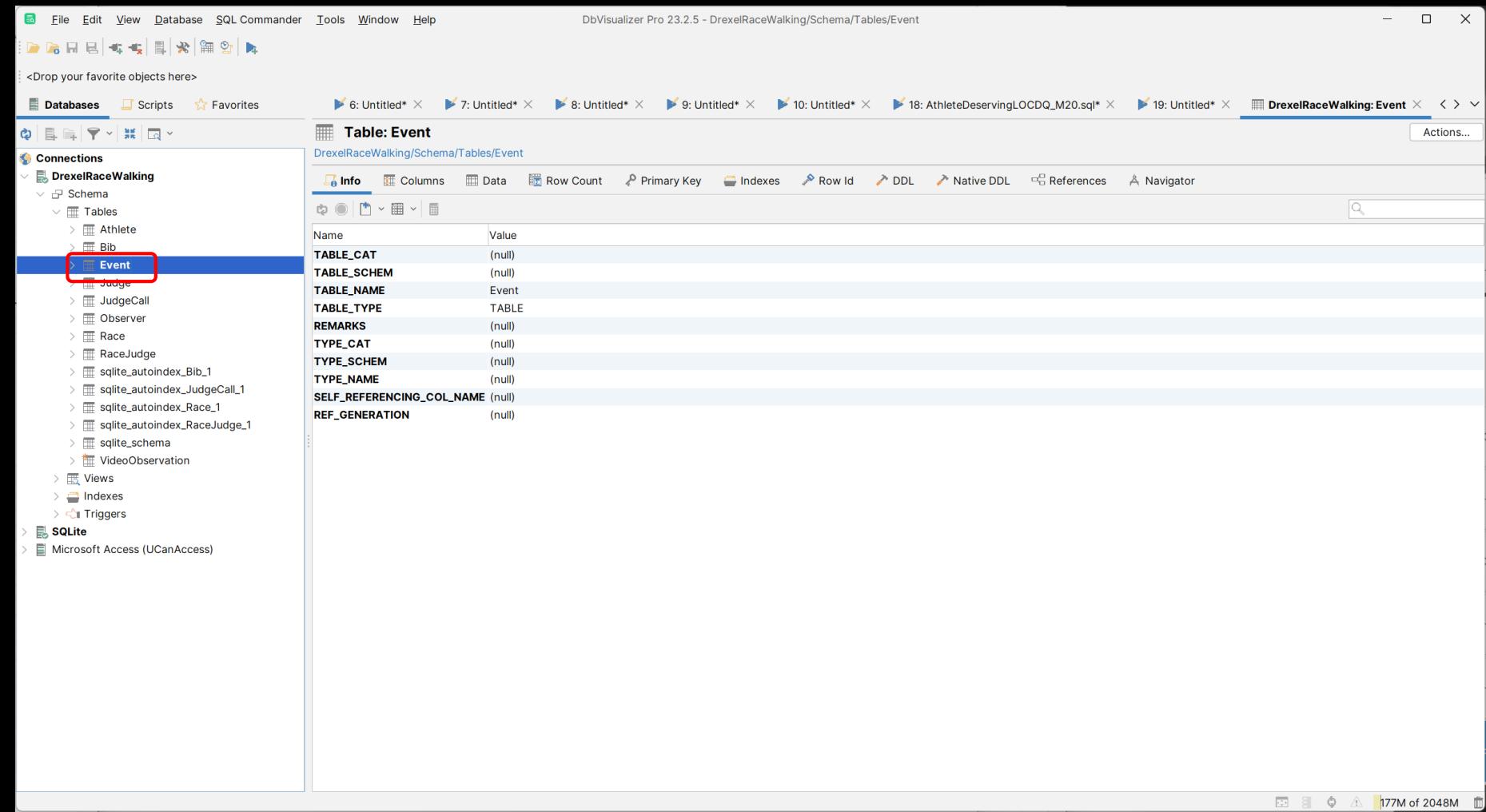


# Inserting Data - GUI

Inserting data manually via DBVisualizer is a simple process

Let's add data to the *Event* table.

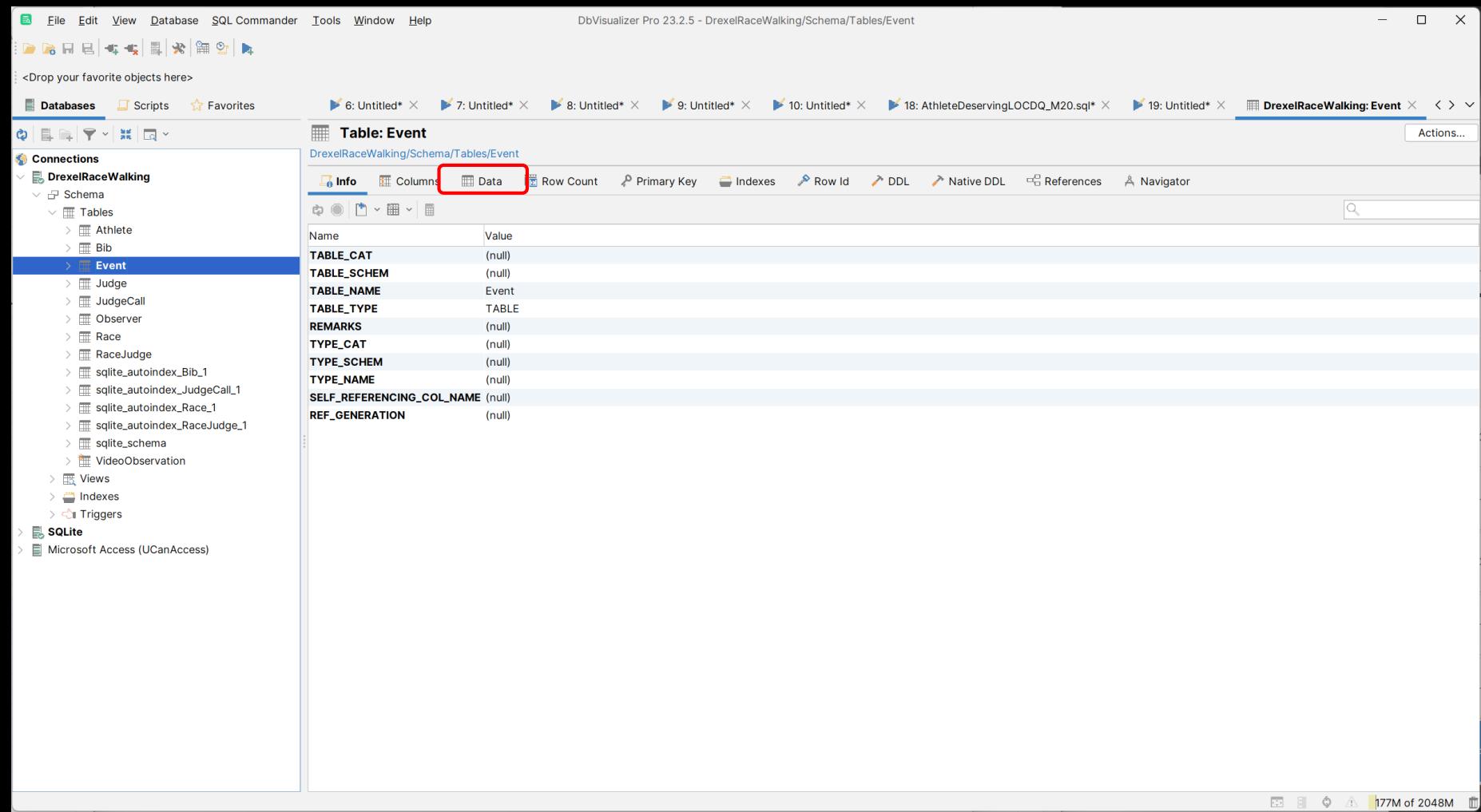
Double Click on the *Event* table:



# Inserting Data - GUI

Let's add data to the *Event* table.

Click on the *Data* tab:



The screenshot shows the DbVisualizer Pro interface. The title bar reads "DbVisualizer Pro 23.2.5 - DrexelRaceWalking/Schema/Tables/Event". The left sidebar shows a tree view of database connections, with "DrexelRaceWalking" selected. Under "Tables", the "Event" table is selected and highlighted with a blue background. The main pane displays the "Table: Event" details. A red box highlights the "Data" tab in the top navigation bar, which is currently active. The data grid below shows the following table structure:

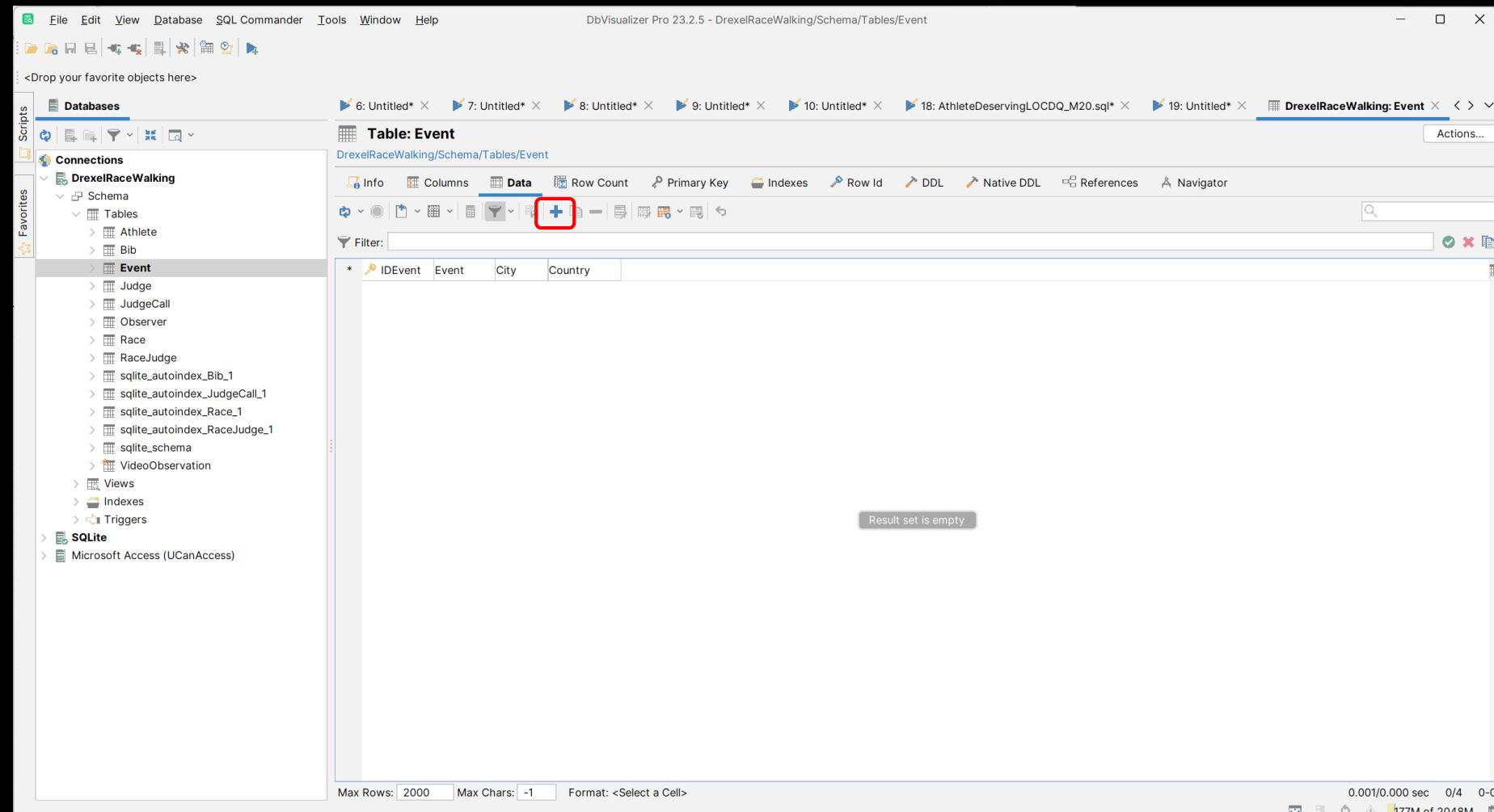
Name	Value
TABLE_CAT	(null)
TABLE_SCHEMA	(null)
TABLE_NAME	Event
TABLE_TYPE	TABLE
REMARKS	(null)
TYPE_CAT	(null)
TYPE_SCHEMA	(null)
TYPE_NAME	(null)
SELF_REFERENCING_COL_NAME	(null)
REF_GENERATION	(null)

# Inserting Data - GUI

Let's add data to the *Event* table.

An empty table appears.

Click on the + icon.



# Inserting Data - GUI

Let's add data to the *Event* table.

A blank row appears, enter your data:

The screenshot shows the DbVisualizer Pro interface version 23.2.5. The left sidebar displays the database connections and tables. The main area is titled "Table: Event" and shows the "Data" tab selected. A single row is present in the data grid, with all four columns (IDEvent, Event, City, Country) containing "(null)" values. This row is highlighted with a red border. The status bar at the bottom indicates "Max Rows: 2000" and "Format: <Select a Cell>".

IDEvent	Event	City	Country
1	(null)	(null)	(null)

# Inserting Data - GUI

Let's add data to the *Event* table.

The data is entered, but it is not saved:

The screenshot shows the DbVisualizer Pro interface version 23.2.5. The main window title is "DbVisualizer Pro 23.2.5 - DrexelRaceWalking/Schema/Tables/Event\*". The left sidebar displays the database schema, including tables like RaceJudge, SQLite\_autoindex\_Bib\_1, and Event. The central pane shows the "Table: Event" with the following data:

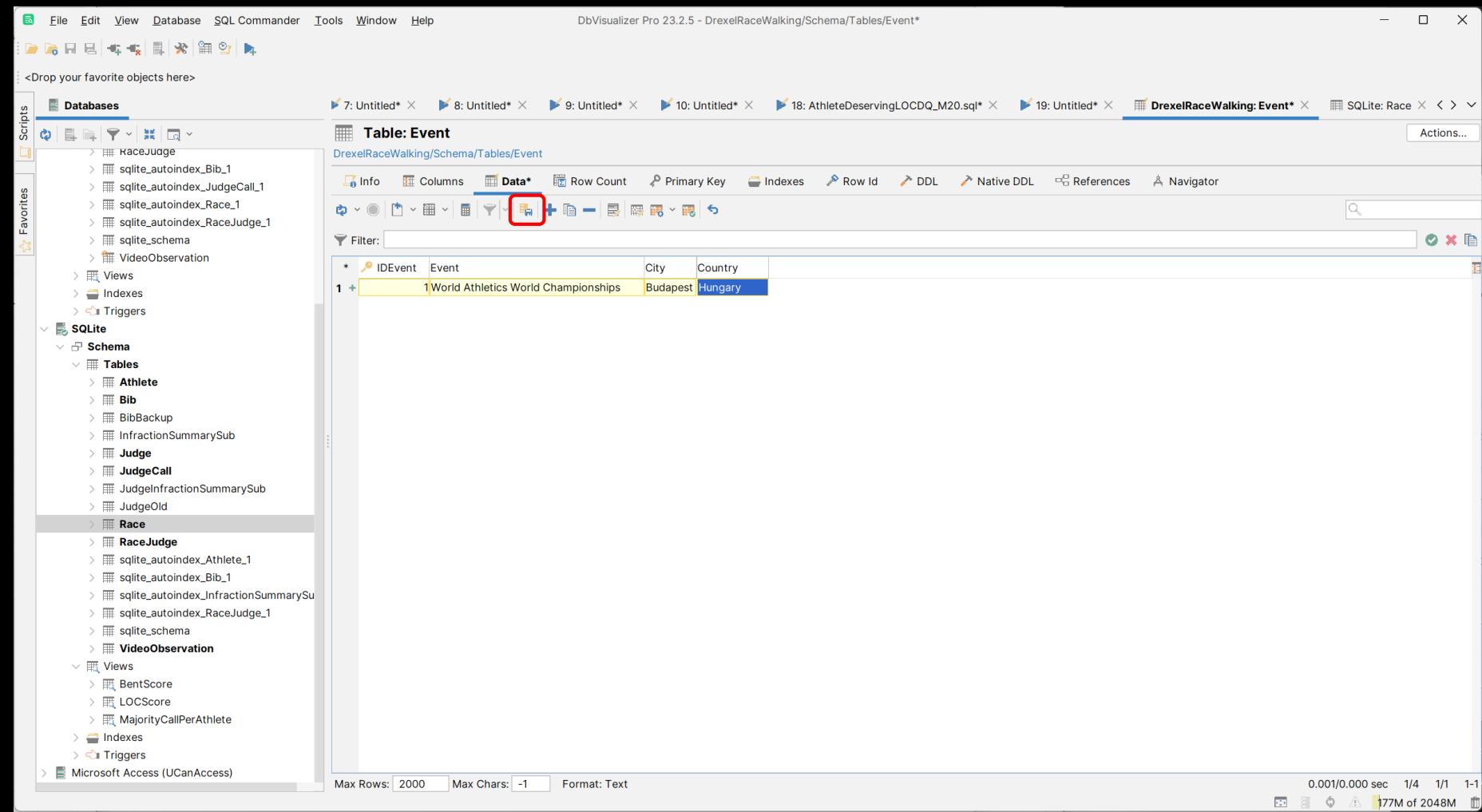
IDEvent	Event	City	Country
1	World Athletics World Championships	Budapest	Hungary

The row with ID 1 is highlighted with a red border. The "Data" tab is selected in the top navigation bar. The status bar at the bottom indicates "Max Rows: 2000 Max Chars: -1 Format: Text" and "0.001/0.000 sec 1/4 1/1 1-1".

# Inserting Data - GUI

Let's add data to the *Event* table.

Click on the *Save* icon:



The screenshot shows the DbVisualizer Pro interface version 23.2.5. The left sidebar displays a tree view of database objects under 'Databases' and 'SQLite Schema'. The main window shows the 'Event' table with one row inserted. The 'Data' tab is selected. A red box highlights the save icon (a green circle with a white checkmark) in the toolbar above the data grid. The data grid shows the following row:

IDEvent	Event	City	Country
1	World Athletics World Championships	Budapest	Hungary

At the bottom of the interface, there are status bars for 'Max Rows: 2000', 'Max Chars: -1', 'Format: Text', and performance metrics: '0.001/0.000 sec', '1/4', '1/1', and '1-1'.

# Inserting Data - GUI

Let's add data to the *Race* table.

We have 4 races, we enter them the same way

Let's see what happens though if we make mistakes. Enter:

The screenshot shows the DbVisualizer Pro interface with the 'Race' table selected. The 'Data' tab is active. A red arrow points from the text 'Note, the incorrect date format. 😞' to the 'RaceDate' column of the first row, which contains the value '2/24/2023'. A red box highlights the 'RaceDate' column header. The table structure is as follows:

IDRace	IDEvent	RaceDate	StartTime	Distance	DistanceUnits	Gender
1	1	2/24/2023	7:00 AM	35	km	(null)

The left sidebar shows the database schema with the 'Race' table selected. The bottom status bar indicates 'Max Rows: 2000' and 'Format: Text'.

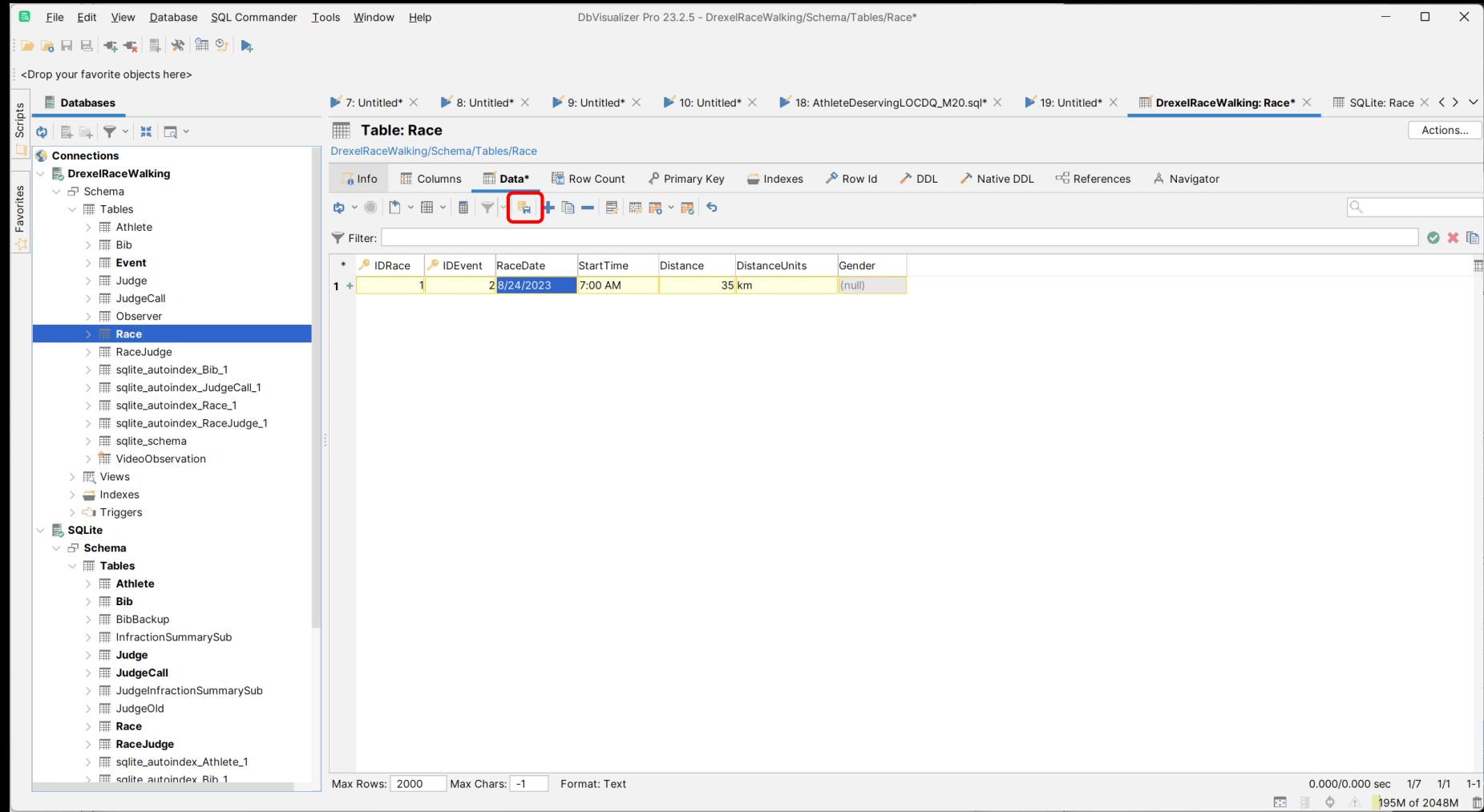
Note, the incorrect  
date format. 😞

# Inserting Data - GUI

What's wrong with the data?

- 1) There is no *IDEvent* in the *Event* table where *IDEvent* = 2
- 2) *Gender* is a required field

Click the save icon anyway:



The screenshot shows the DbVisualizer Pro interface. On the left, the navigation pane displays the database schema, including the 'Race' table under the 'DrexelRaceWalking' connection. The main area shows the 'Race' table data grid. A single row is present with the following values:

IDRace	IDEvent	RaceDate	StartTime	Distance	DistanceUnits	Gender
1	1	2023-08-24	7:00 AM	35	km	(null)

A red box highlights the save icon (a blue floppy disk) in the toolbar above the data grid.

# Inserting Data - GUI

The follow error appears:

The screenshot shows the DbVisualizer Pro interface. At the top, the title bar reads "DbVisualizer Pro 23.2.5 - DrexelRaceWalking/Schema/Tables/Race\*". Below the title bar is a toolbar with various icons. A message dialog box is open, titled "Save Result(s)". The message inside says: "There were problems saving to the database table. Review the list, do the necessary changes and try again." A note below it states: "Note: DbVisualizer use bind variables when executing these statements. The effect of this is that the SQL listed below may not be 100% compliant with data formats for the target database i.e. they may fail to execute in for example the SQL Commander." The main window shows a table of errors. The first row has columns "Row", "DML", "Message", and "SQL". The "Message" column contains the error: "1 + INSERT org.sqlite.SQLiteException: [SQLITE\_CONSTRAINT\_NOTNULL] A NOT NULL constraint failed (NOT NULL constraint failed: Race.Gend...)" and the "SQL" column contains the corresponding SQL statement: "INSERT INTO "Race" ("IDRace", "IDEEvent", "RaceDate", "Gend...")". At the bottom right of the error dialog are buttons for "Key Column(s)..." and "Close". In the bottom left corner of the main window, there is a dark overlay. The bottom right corner of the main window shows performance metrics: "0.000/0.000 sec", "1/7", "1/1", "1-1", "195M of 2048M", and a small progress bar.

Row	DML	Message	SQL
1	+ INSERT	org.sqlite.SQLiteException: [SQLITE_CONSTRAINT_NOTNULL] A NOT NULL constraint failed (NOT NULL constraint failed: Race.Gend...)	INSERT INTO "Race" ("IDRace", "IDEEvent", "RaceDate", "Gend...")

Key Column(s)... Close

SQLite Schema Tables Athlete Bib BibBackup InfractionSummarySub Judge JudgeCall JudgeInfractionSummarySub JudgeOld Race RaceJudge sqlite\_autoindex\_Athlete\_1 sqlite\_autoindex\_Rib\_1

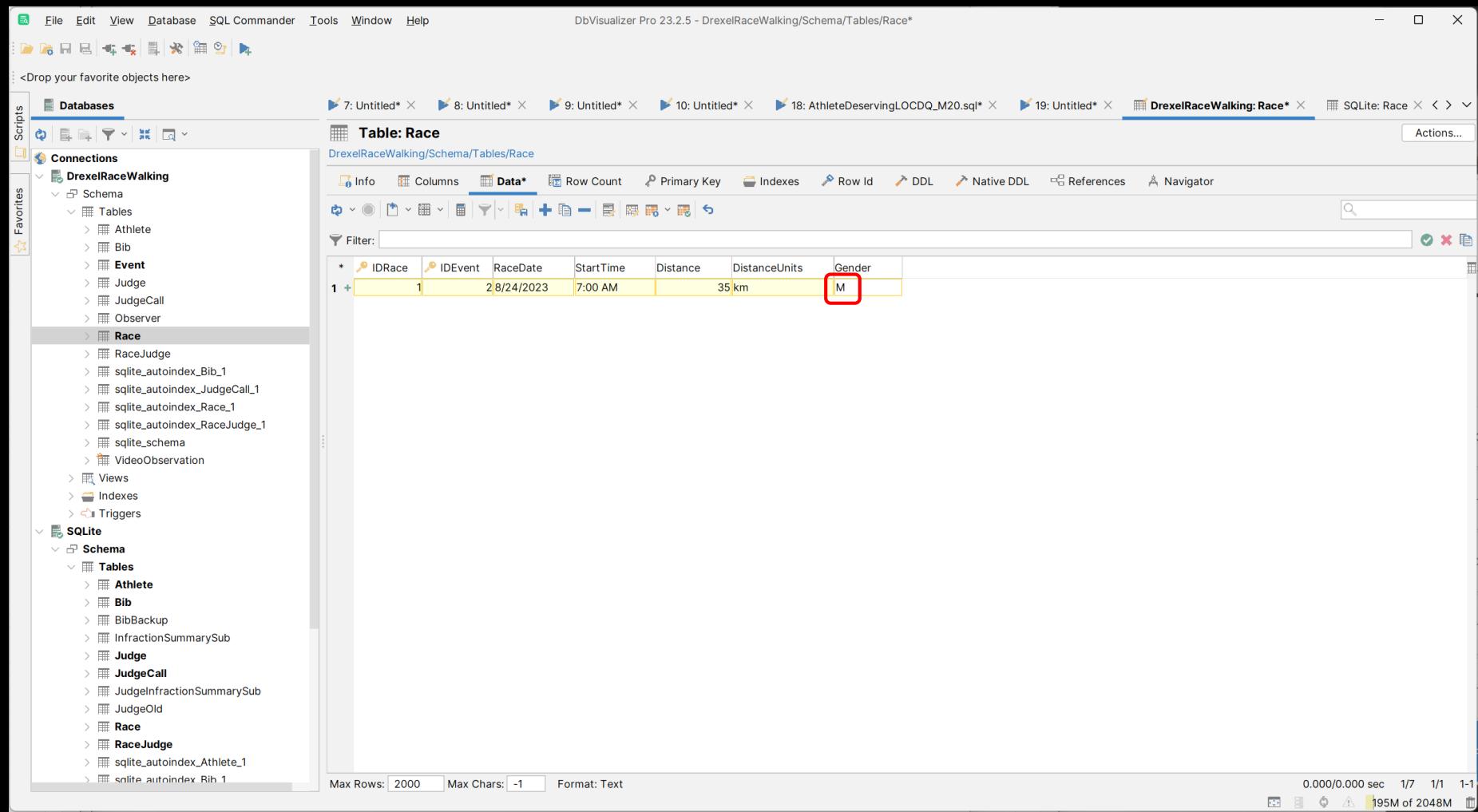
Max Rows: 2000 Max Chars: -1 Format: Text

0.000/0.000 sec 1/7 1/1 1-1 195M of 2048M

# Inserting Data - GUI

Hit *Close* and enter the *Gender* as "M"

Hit the *Save* icon again:



The screenshot shows the DbVisualizer Pro interface with the 'Race' table selected. The data grid displays the following information for one row:

IDRace	IDEvent	RaceDate	StartTime	Distance	DistanceUnits	Gender
1	1	28/24/2023	7:00 AM	35	km	M

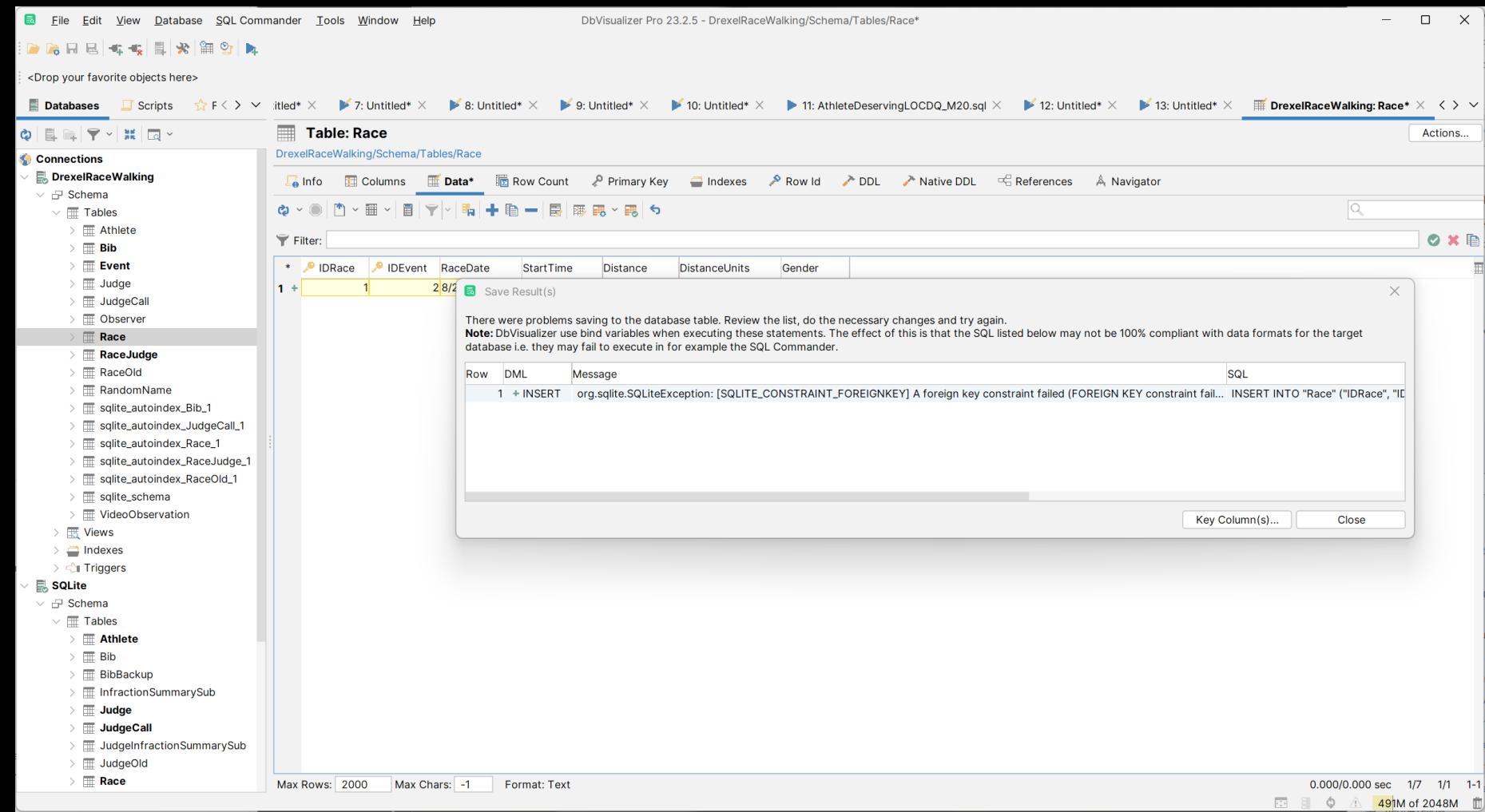
A red box highlights the 'Gender' cell containing 'M'. The left sidebar shows the database schema with the 'Race' table selected.

# Inserting Data - GUI

Hit **Close** and enter the *Gender* as "M"

We should get an error because the Foreign Keys are violated.

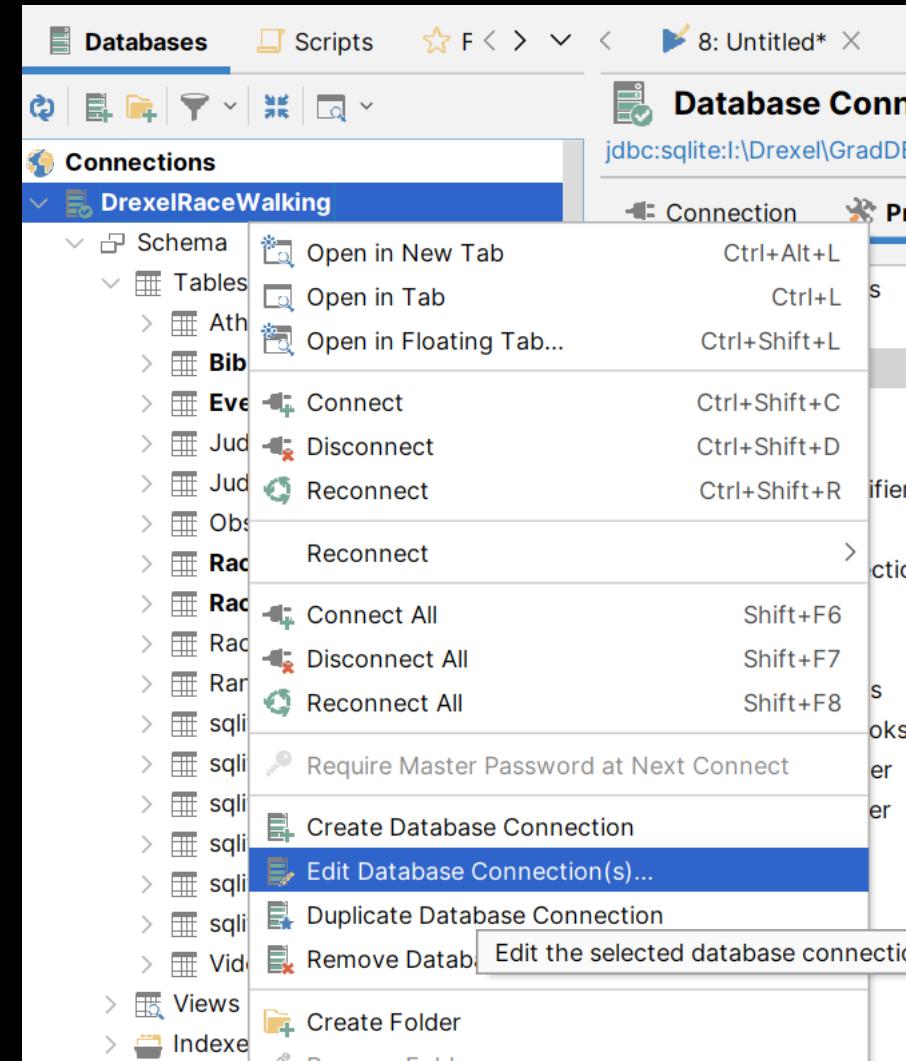
This is a good warning. It prevents accidentally entered data errors.



# Inserting Data - GUI

If the error doesn't appear, it's likely that enforcing Foreign Keys is not on by default.

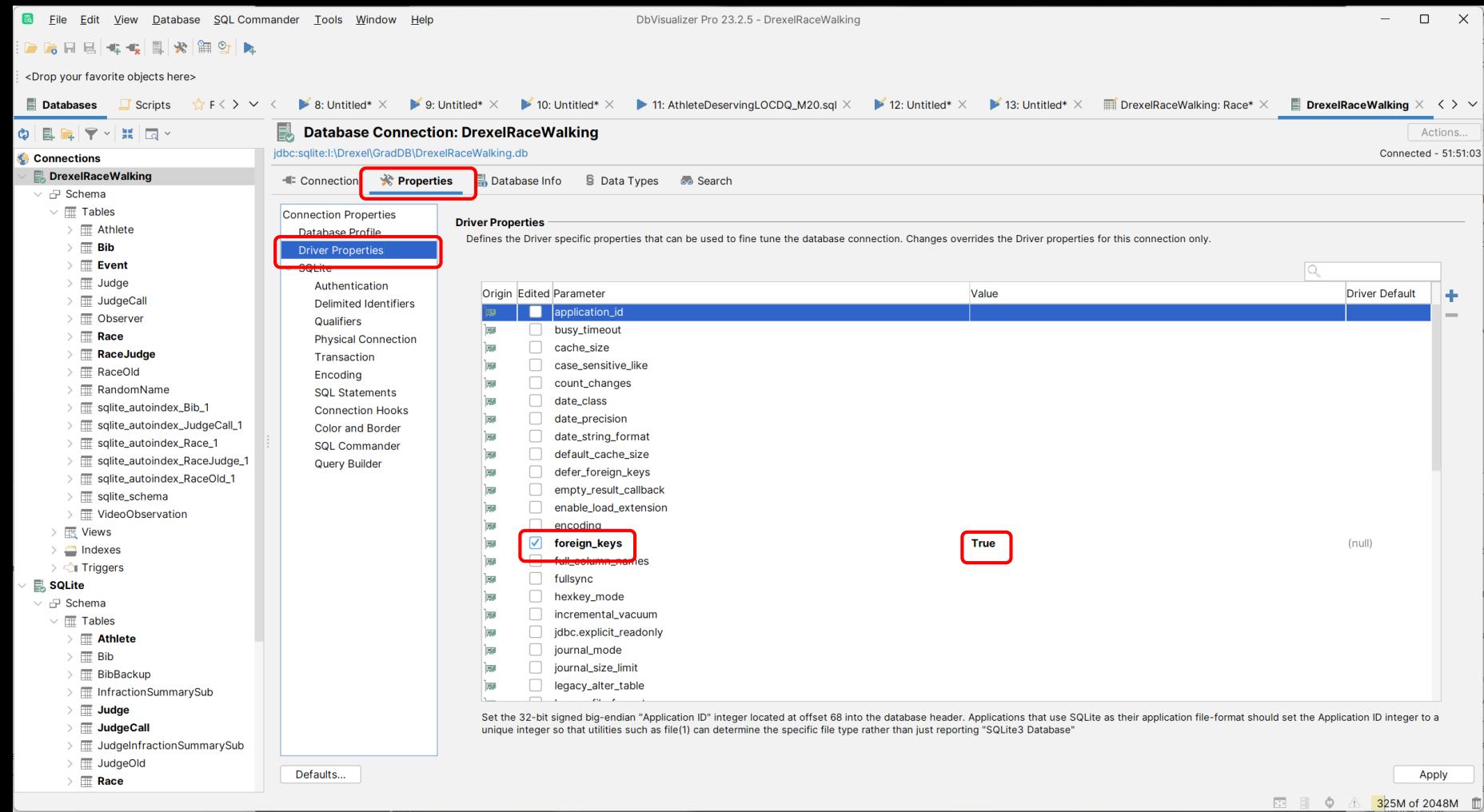
Right-click on the connection, then click on *Edit Database Connections (s)...*



# Inserting Data - GUI

Click on *Properties* and then *Driver Properties*.

Enter True for the value of *foreign\_keys*.



# Inserting Data - GUI

Sometimes it is required to import data from an external source, like a spreadsheet:

The screenshot shows a Microsoft Excel window titled "VisualObservation.xlsx". The ribbon is visible at the top with tabs for File, Home, Insert, Draw, Page Layout, Formulas, Data, Review, View, Automate, Help, Acrobat, and Comments. The Home tab is selected. The main area displays a data table with approximately 40 rows of data. The columns are labeled A through M. Column A contains IDs ranging from 1 to 1360. Columns B through M contain various numerical and categorical values. The data includes observations such as Bib Number, Race ID, Observation ID, and various location and angle measurements. The "Clipboard" icon in the ribbon indicates that data has been copied or is being pasted. The status bar at the bottom shows "Sheet Sheet-SQL" and "Ready Accessibility: Good to go".

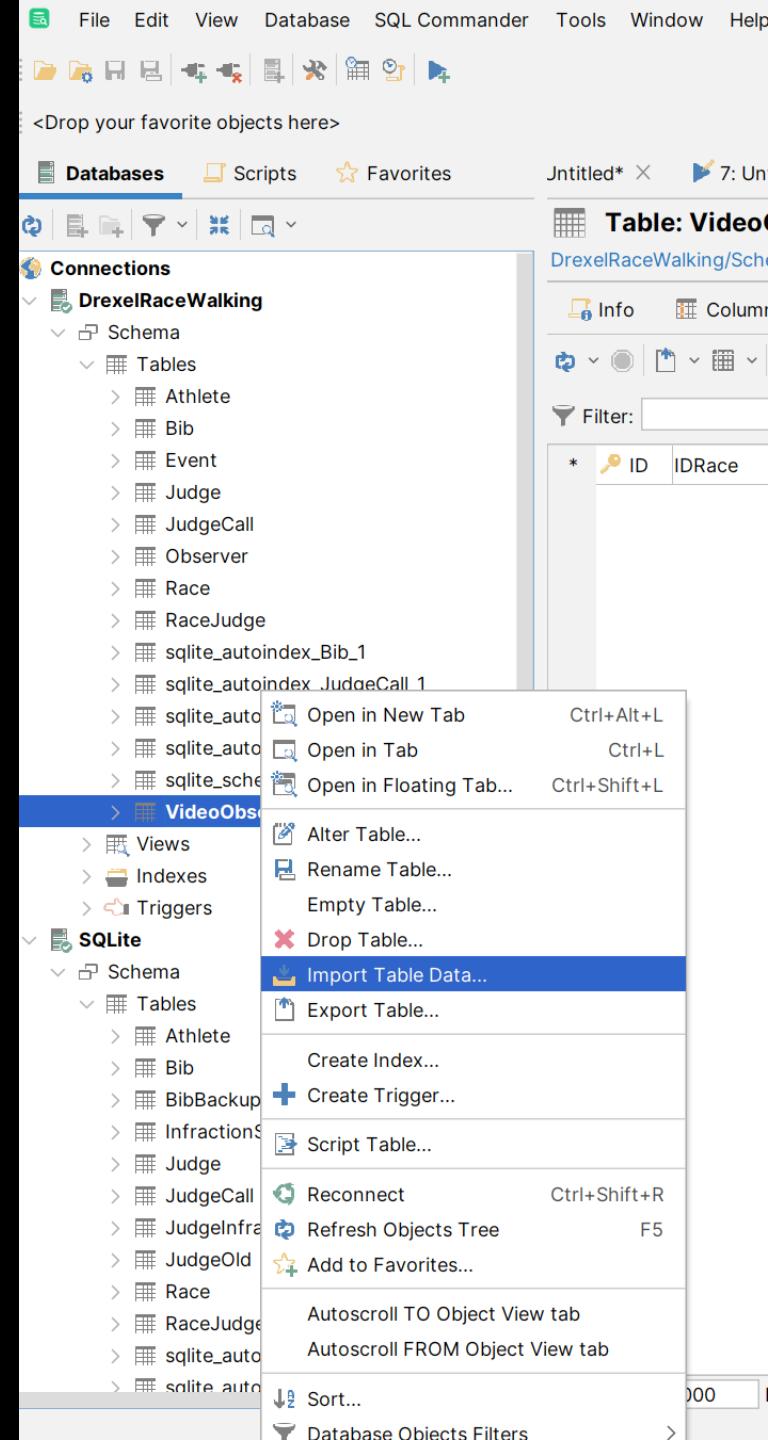
1	A	B	C	D	E	F	G	H	I	J	K	L	M
1	ID	IDRace	BibNumber	IDObserve	LOCAverage	LOC1	LOC2	LOC3	LOC4	BentKneeAngle	Comment	VideoFile	TOD
2	59	1	167	5	30.5					0	C1007	43296000	
3	20	1	182	3	4					0	C1311	53246000	
4	29	1	120	5	2					176 175.7	C1317	53777000	
5	19	1	175	3	8.3					175 175	C1310	53223000	
6	14	1	148	3	8.333					173	C1305	53036000	
7	149	1	336	1	45.8					0	C1014	44084000	
8	1	1	127	3	62.5					0	C1293	52643000	
9	889	3	103	1	42.2 6.5	3.25	6.25	4.25		0	C0960	47434000	
10	134	1	357	1	54.2					0	C1012	44052000	
11	135	1	335	1	45.8					0	C1012	44052000	
12	45	1	379	5	36.5					0	C1007	43296000	
13	32	1	182	5	8.33					176 175.6	C1320	53992000	
14	34	1	175	5	18					175 15.4	C1292	52597000	
15	73	1	169	1	45.8					-1	C1009	43426000	
16	600	2	348	3	20.3 0.75	3	3	3		0	C0835	59465000	
17	936	3	100	1	38.5 4.25	4.75	4.5	5		177 177	C0952	46712000	
18	1215	1	378	3	28.1 4	3.5	3.5	2.5		176 175.7	C1093	46812000	
19	880	3	100	1	41.1 5	5.5	5	4.25		176 175.5	C0963	47596000	
20	1253	1	372	1	39.6 7	5	6.5	0.5		175 175.3	C1105	47218000	
21	1588	1	328	1	41.3 5.3	4.5	5	5		175 175.3	C1260	51672000	
22	575	2	377	1	43.8 5	5.5	5	5.5		175 174.9	C0825	59270000	
23	1275	1	127	1	50.5 6	6.25	6	6		175 174.8	C1117	47466000	
24	1412	1	376	1	32.3 4	3.5	4	4		175 174.5	C1174	49287000	
25	1582	1	167	1	36.5 4.5	4	4	5		174 174.3	C1260	51672000	
26	1152	3	127	3	60.4 7.75	6.75	0	0		174 174.3	C0920	44188000	
27	1342	1	166	1	38.5 3	5.5	4	6		174 174.2	C1149	48517000	
28	344	1	345	5	59.4 7.25	7	6.25	8		174 174.2	C1048	45641000	
29	1138	3	127	3	57.3 7	7	6.25	7.25		174 174	C0921	44462000	
30	1222	1	127	3	48.4 5.75	6	5.25	6.25		174 174	C1096	46914000	
31	228	1	110	1	40.1					174 173.9	C1027	44735000	
32	152	1	110	1	43.2					174 173.8	C1015	44165000	
33	352	1	167	5	42.7 4	5	6	5.5		174 173.5	C1054	45815000	
34	1139	3	167	3	47.4 6	5.5	5.5	5.75		174 173.5	C0921	44462000	
35	1228	1	125	3	47.9 6	5.5	0	0		173 173.4	C1099	47033000	
36	1402	1	371	1	34.4 5	4	4	3.5		173 173.3	C1169	49076000	
37	1560	1	127	1	60.9 6.75	7.5	7.5	7.5		173 173.3	C1245	51187000	
38	1360	1	167	1	22 2 1	1	0	0		172 172 2	C1150	48576000	

# Inserting Data - GUI

Another way to import data is to use the *Import Table Data* feature.

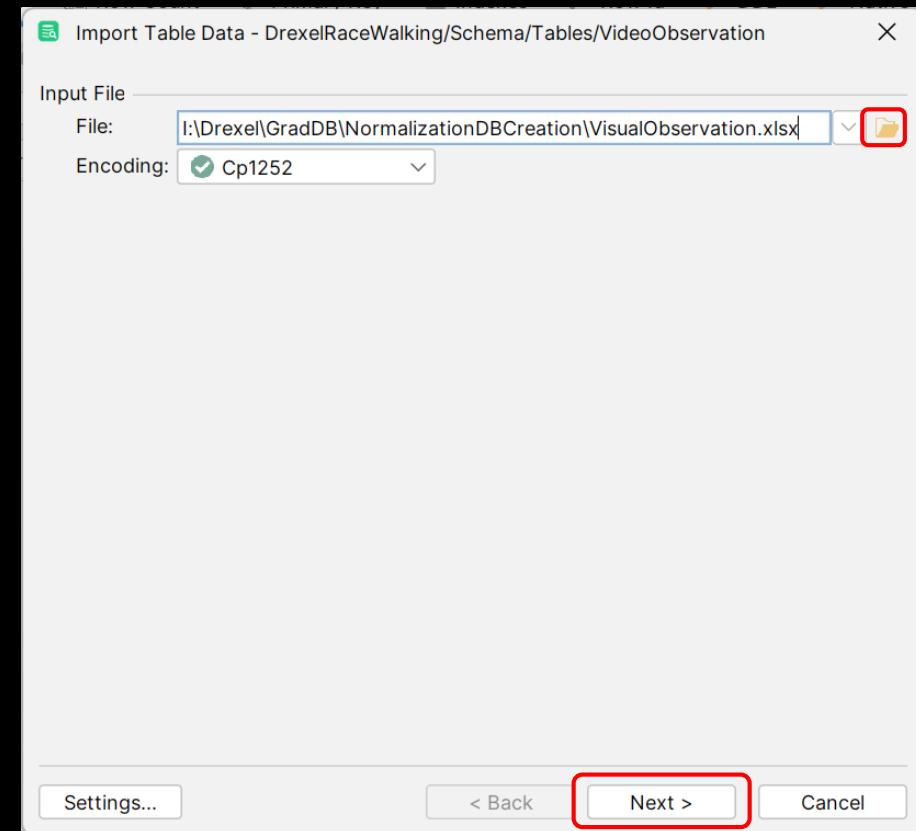
Right-click on the destination table.

Select *Import Table Data* from the popup menu:



# Inserting Data - GUI

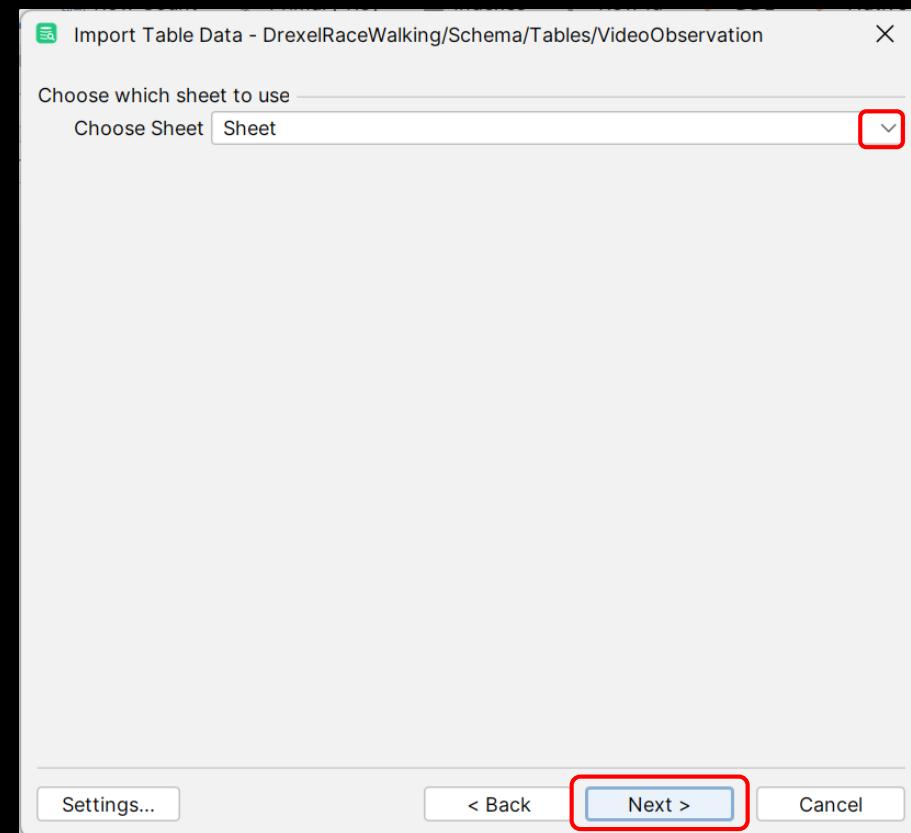
Select the file you wish to import and click next:



# Inserting Data - GUI

In this case, there is only one sheet in the spreadsheet, but we could select the other sheets if they existed.

Click on the *Next* button.

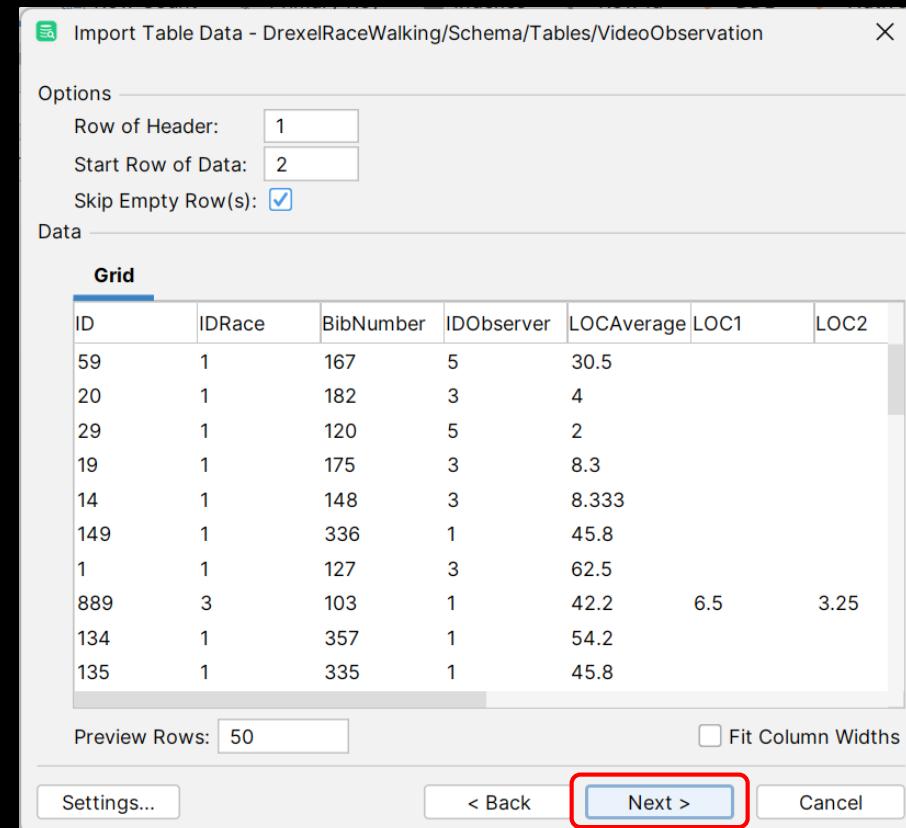


# Inserting Data - GUI

We get a preview of what the data looks like.

We can even specify the row for headers and where the data starts.

Click on the *Next* button.



# Inserting Data - GUI

We can set formats and change data types if needed.

Most in this case, are defaulted incorrectly:

Import Table Data - DrexelRaceWalking/Schema/Tables/VideoObservation

Data Formats

Date: yyyy-MM-dd 2023-11-12  
Time: HH:mm:ss 11:17:28  
Timestamp: yyyy-MM-dd HH:mm:ss 2023-11-12 11:17:28  
Number Separators: Grouping , Decimal .  
Boolean True ie, yes, 1, on False ls, no, 0, off  
Null Value Text (null)

Data

Grid

ID	IDRace	BibNumber	IDObserver	LOCAverage	LOC1	LOC2
Decim...	Decim...	Decim...	Decim...	Decim...	String	Stri
59	1	167	5	30.5		
20	1	182	3	4		
29	1	120	5	2		
19	1	175	3	8.3		
14	1	148	3	8.333		

Preview Rows: 50  Fit Column Widths

Settings... < Back Next > Cancel

# Inserting Data - GUI

Click on each field and correct the data types.

ID: Number

IDRace: Number

BibNumber: Number

IDObserver: Number

LOC1: Decimal

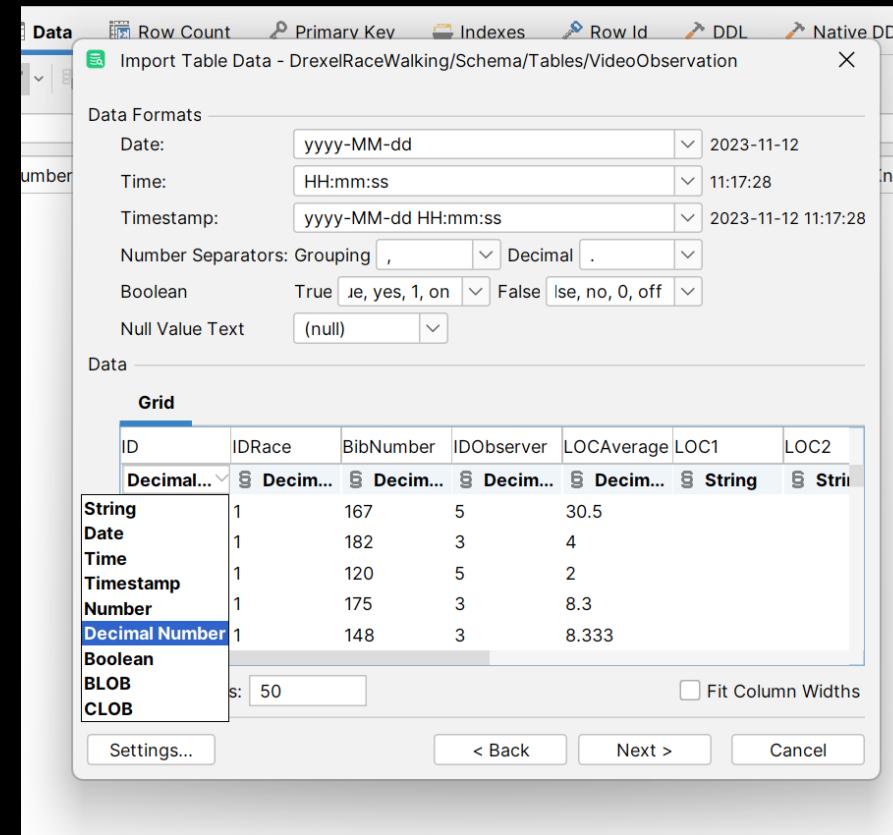
LOC2: Decimal

LOC3: Decimal

LOC4: Decimal

TOD: String (More on this later)

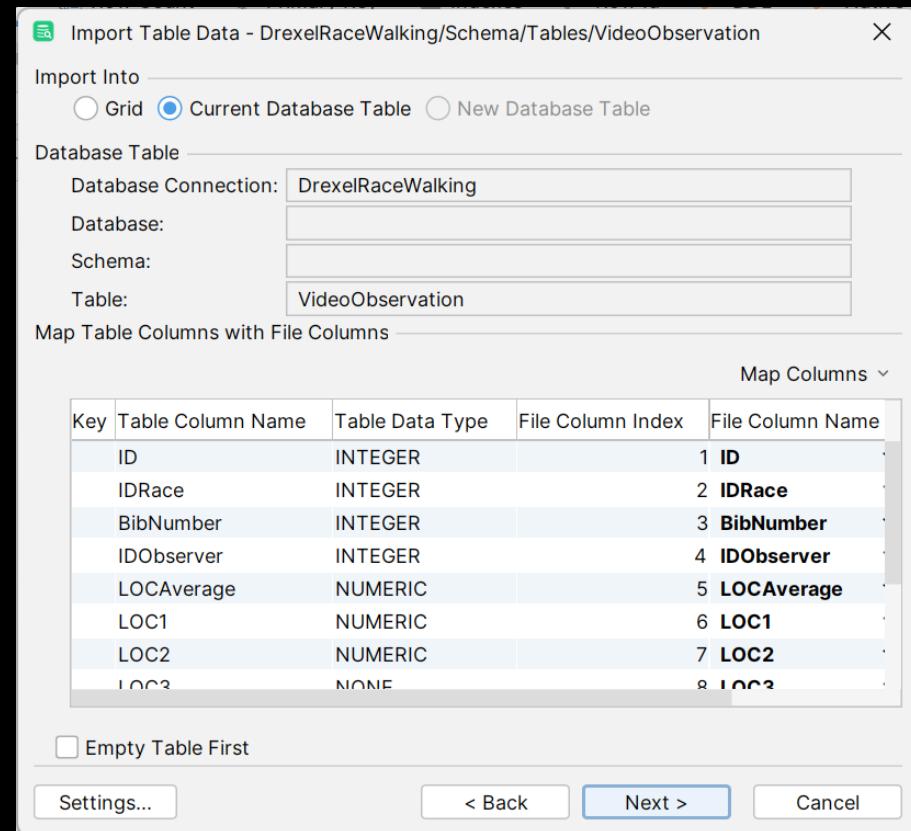
Click on the **Next** button.



# Inserting Data - GUI

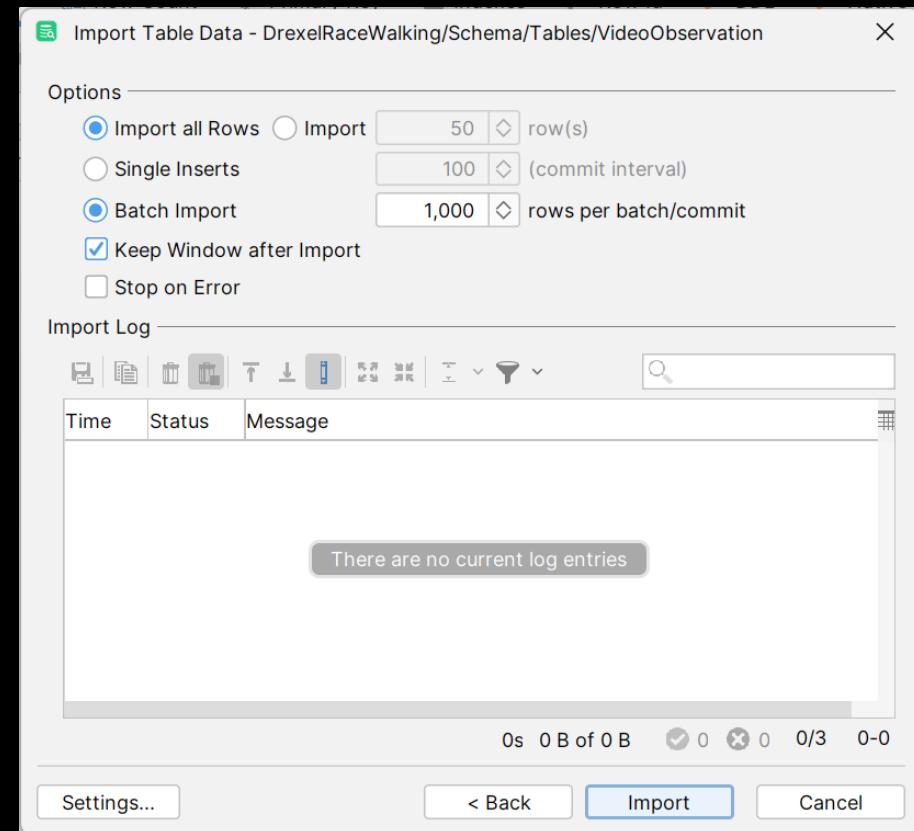
Confirm your changes

Click on the **Next** button.



# Inserting Data - GUI

Click on the *Import* button.



# Inserting Data - GUI

To see your changes, click the *Refresh* icon.

The screenshot shows the DbVisualizer Pro interface version 23.2.5. The main window title is "DbVisualizer Pro 23.2.5 - DrexelRaceWalking/Schema/Tables/VideoObservation". The left sidebar displays the "Connections" tree, which includes a "DrexelRaceWalking" connection under "Schema" and an "SQLite" connection under "Schema". The "VideoObservation" table is selected under the DrexelRaceWalking connection. The right pane shows the "Table: VideoObservation" details, specifically the "Data" tab. A red box highlights the refresh icon (a circular arrow) in the toolbar above the data grid. The data grid itself is empty, with a message "Result set is empty" displayed below it. The bottom status bar shows "Max Rows: 2000 Max Chars: -1 Format: <Select a Cell>".

# Inserting Data - GUI

Your imported data appears:

The screenshot shows the DbVisualizer Pro interface. The left sidebar displays the database connections and schema, specifically the 'DrexelRaceWalking' database under the 'Tables' section. The main window shows the 'VideoObservation' table with 15 rows of data. The table has columns: \*, ID, IDRace, BibNumber, IDObserver, LOCAverage, LOC1, LOC2, LOC3, LOC4, and KneeAng. Row 6 is currently selected, highlighting the values: ID=6, IDRace=1, BibNumber=117, IDObserver=3, LOCAverage=39, LOC1=null, LOC2=null, LOC3=null, LOC4=null, and KneeAng=null. The bottom status bar indicates the following: Max Rows: 2000, Max Chars: -1, Number: Unformatted, 0.001/0.007 sec, 1630/13, 1/1, 1-15, and 129M of 2048M.

*	ID	IDRace	BibNumber	IDObserver	LOCAverage	LOC1	LOC2	LOC3	LOC4	KneeAng
1	1	1	127	3	62.5	(null)	(null)	(null)	(null)	
2	2	1	166	3	33.7	(null)	(null)	(null)	(null)	
3	3	1	151	3	19.4	(null)	(null)	(null)	(null)	
4	4	1	164	3	30.7	(null)	(null)	(null)	(null)	
5	5	1	119	3	37	(null)	(null)	(null)	(null)	
6	6	1	117	3	39	(null)	(null)	(null)	(null)	
7	7	1	158	3	0	(null)	(null)	(null)	(null)	
8	8	1	378	3	14.6	(null)	(null)	(null)	(null)	
9	9	1	185	3	28	(null)	(null)	(null)	(null)	
10	10	1	107	3	0	(null)	(null)	(null)	(null)	
11	11	1	159	3	28.1	(null)	(null)	(null)	(null)	
12	12	1	125	3	31.3	(null)	(null)	(null)	(null)	
13	13	1	131	3	0	(null)	(null)	(null)	(null)	
14	14	1	148	3	8.333	(null)	(null)	(null)	(null)	
15	15	1	174	3	17.7	(null)	(null)	(null)	(null)	

Max Rows: 2000 Max Chars: -1 Number: Unformatted 0.001/0.007 sec 1630/13 1/1 1-15 129M of 2048M

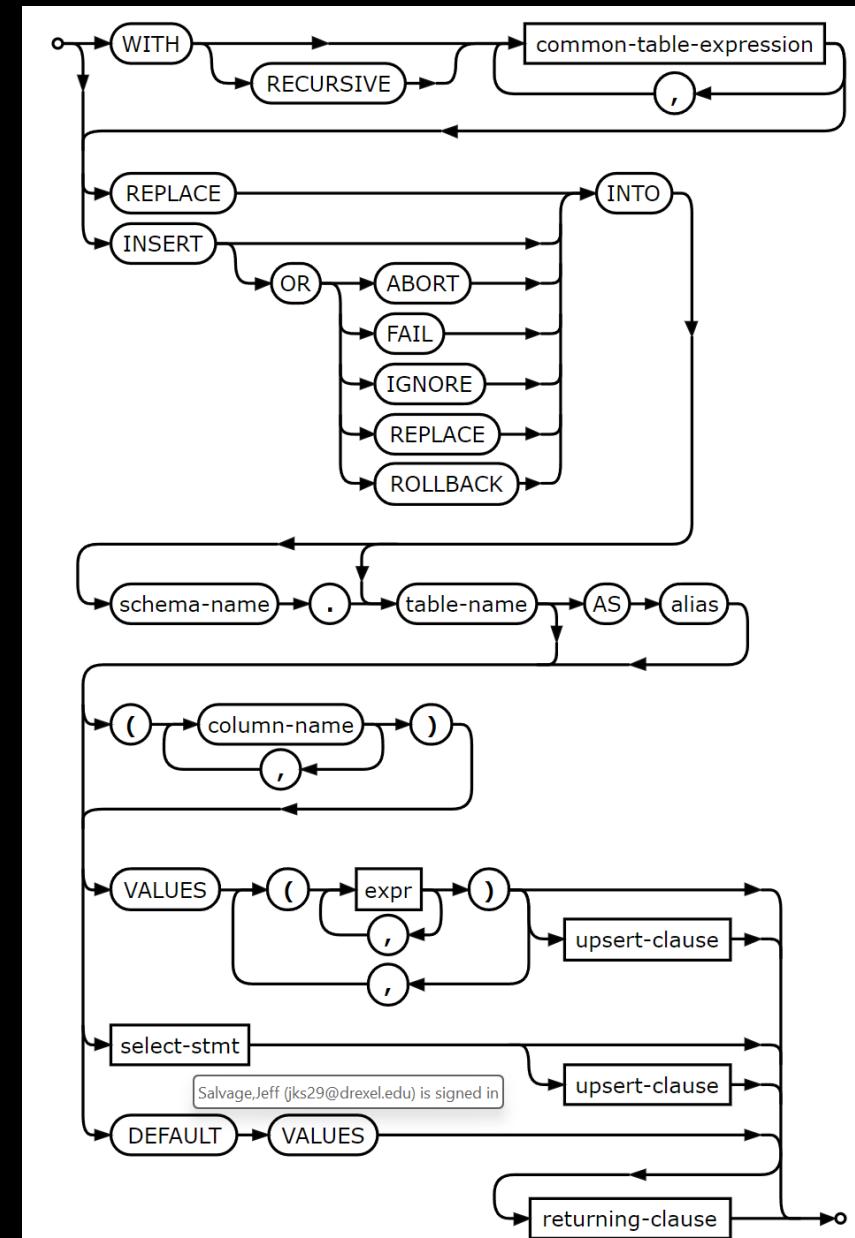
# Inserting Data – Via Code

The SQL INSERT statement can be used to insert data into the database one row at a time:

The SQL INSERT statement has many options.

The basic for entering a list of values is:

```
INSERT INTO table VALUES(...);
```



For more information see:

[https://www.sqlite.org/lang\\_insert.html](https://www.sqlite.org/lang_insert.html)

# Inserting Data – Via Code

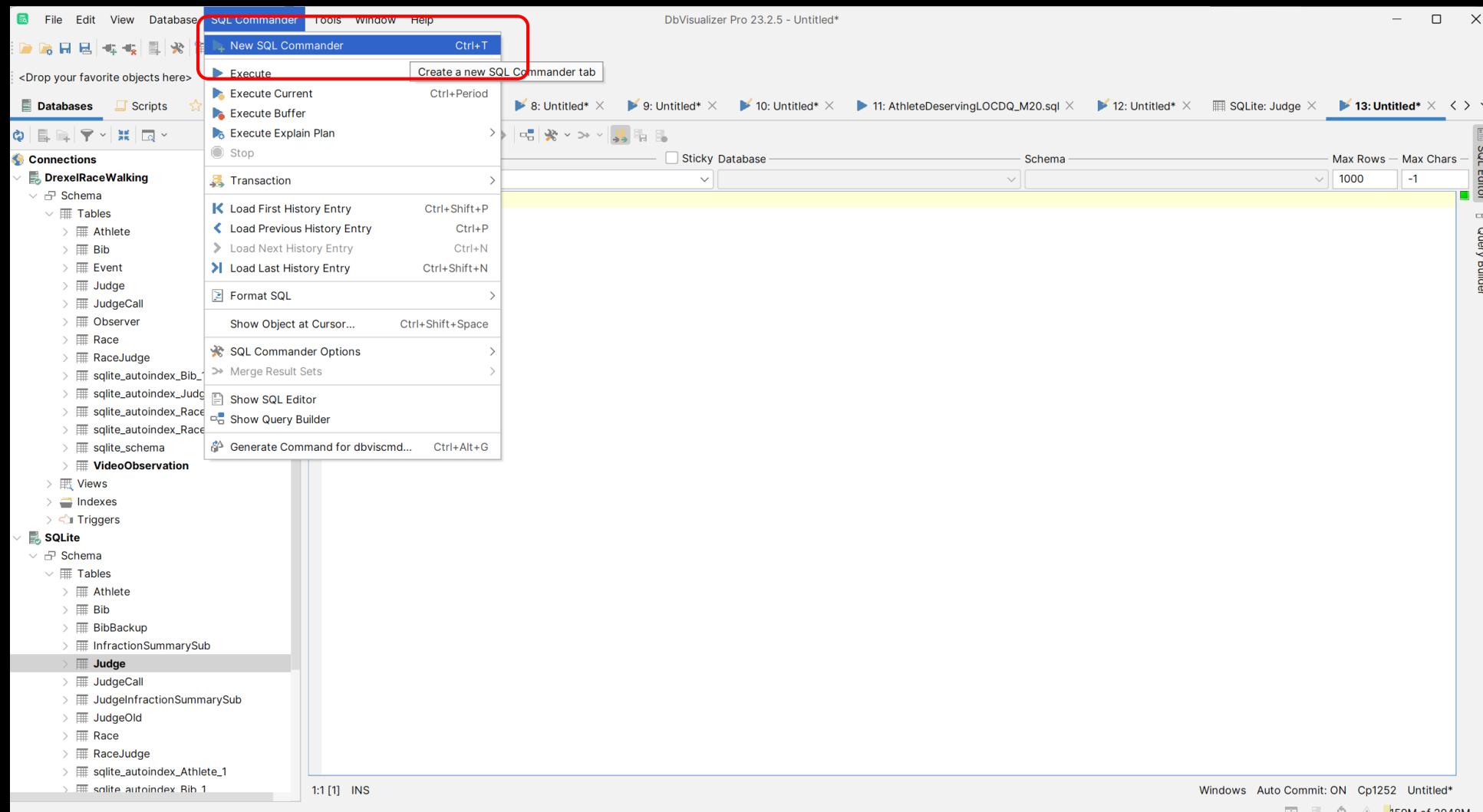
Let's create 8 `INSERT` statements to load the *Judge* table:

```
INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES (1, 'FRA', 'Jean-Pierre', 'DAHM');  
INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES (2, 'POR', 'José', 'DIAS');  
INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES (3, 'FIN', 'Anne', 'FRÖBERG');  
INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES (4, 'CAN', 'Daniel', 'MICHAUD');  
INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES (5, 'ARG', 'Guillermo', 'PERA VALLEJOS');  
INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES (6, 'GRB', 'Ian', 'RICHARDS');  
INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES (7, 'ESP', 'Sergio', 'SOLANA SOR');  
  
INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName)  
VALUES (8, 'HKG', 'Man Chun', 'YEUNG');
```

IDJudge	1	CountryCode	FirstName	LastName
	1	FRA	Jean-Pierre	DAHM
	2	POR	José	DIAS
	3	FIN	Anne	FRÖBERG
	4	CAN	Daniel	MICHAUD
	5	ARG	Guillermo	PERA VALLEJOS
	6	GRB	Ian	RICHARDS
	7	ESP	Sergio	SOLANA SOR
	8	HKG	Man Chun	YEUNG

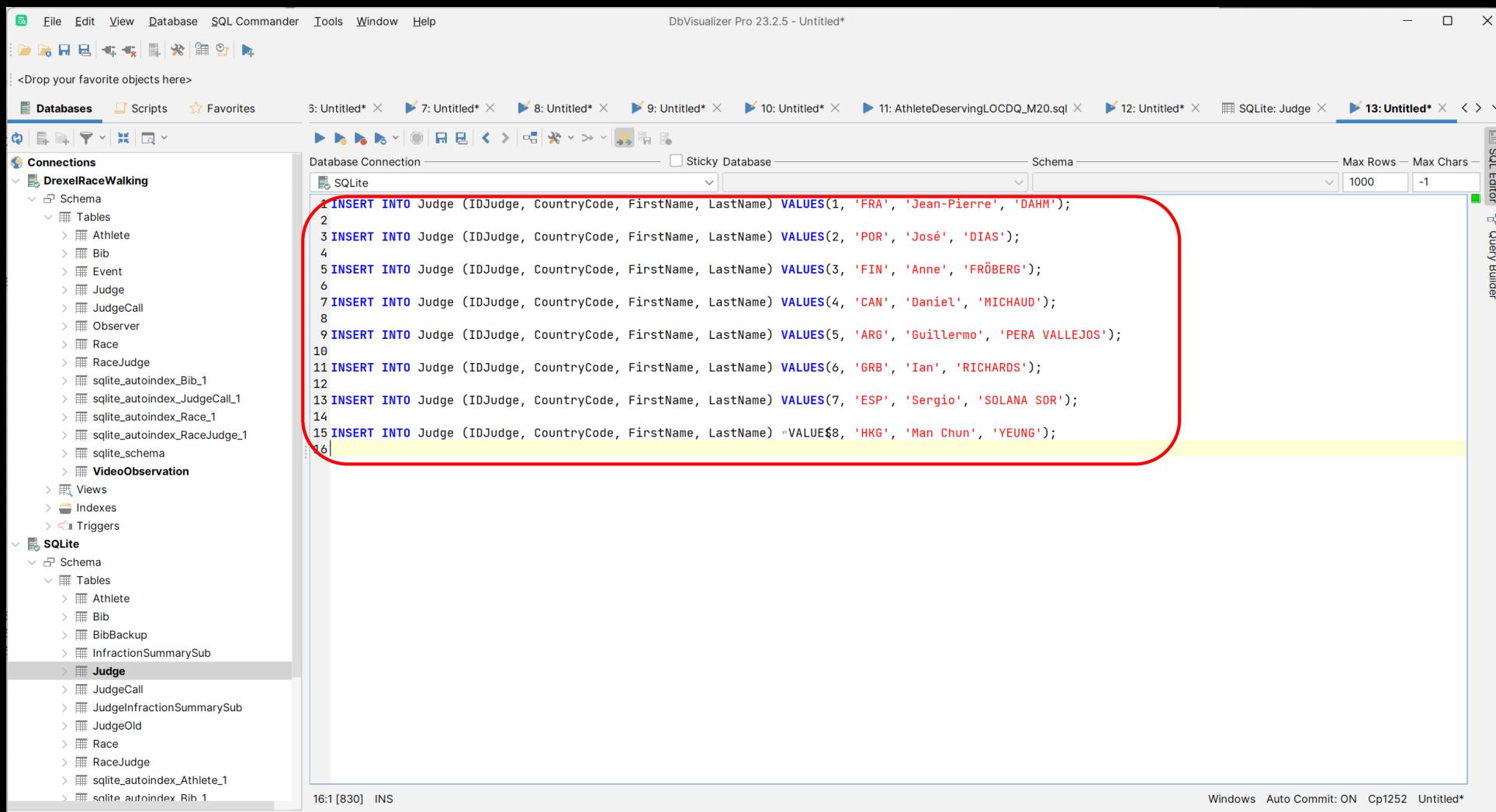
# Inserting Data – Via Code

Open the SQL Commander by clicking on the *SQL Commander* menu and the *New SQL Commander* menu item:



# Inserting Data – Via Code

Enter the new code:



The screenshot shows the DbVisualizer Pro interface. On the left, the Connections pane displays two databases: 'DrexelRaceWalking' and 'SQLite'. Under 'DrexelRaceWalking', the 'Tables' section is expanded, showing 'Athlete', 'Bib', 'Event', 'Judge', 'JudgeCall', 'Observer', 'Race', 'RaceJudge', 'sqlite\_autoindex\_Bib\_1', 'sqlite\_autoindex\_JudgeCall\_1', 'sqlite\_autoindex\_Race\_1', 'sqlite\_autoindex\_RaceJudge\_1', 'sqlite\_schema', 'VideoObservation', 'Views', 'Indexes', and 'Triggers'. Under 'SQLite', the 'Tables' section is expanded, showing 'Athlete', 'Bib', 'BibBackup', 'InfractionSummarySub', and 'Judge'. The 'Judge' table is currently selected. The main window contains a SQL Editor tab titled 'SQLite: Judge'. The code in the editor is:

```
1 INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(1, 'FRA', 'Jean-Pierre', 'DAHM');
2
3 INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(2, 'POR', 'José', 'DIAS');
4
5 INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(3, 'FIN', 'Anne', 'FRÖBERG');
6
7 INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(4, 'CAN', 'Daniel', 'MICHAUD');
8
9 INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(5, 'ARG', 'Guillermo', 'PERA VALLEJOS');
10
11 INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(6, 'GRB', 'Ian', 'RICHARDS');
12
13 INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(7, 'ESP', 'Sergio', 'SOLANA SOR');
14
15 INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(8, 'HKG', 'Man Chun', 'YEUNG');
```

A red oval highlights the first 15 lines of the SQL code. The status bar at the bottom right indicates 'Windows Auto Commit: ON Cp1252 Untitled\*' and '159M of 2048M'.

# Inserting Data – Via Code

The 8 records are inserted into the database.

The screenshot shows the DbVisualizer Pro interface with the following details:

- Connections:** DrexelRaceWalking
- Tables:** Athlete, Bib, Event, Judge, JudgeCall, Observer, Race, RaceJudge, sqlite\_autoindex\_Bib\_1, sqlite\_autoindex\_JudgeCall\_1, sqlite\_autoindex\_Race\_1, sqlite\_autoindex\_RaceJudge\_1, sqlite\_schema, VideoObservation, Views, Indexes, Triggers
- SQL Editor:** An SQL script titled "Untitled\*" containing 15 INSERT INTO statements for the Judge table. The script is as follows:

```
1 INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(1, 'FRA', 'Jean-Pierre', 'DAHM');
2
3 INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(2, 'POR', 'José', 'DIAS');
4
5 INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(3, 'FIN', 'Anne', 'FRÖBERG');
6
7 INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(4, 'CAN', 'Daniel', 'MICHAUD');
8
9 INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(5, 'ARG', 'Guillermo', 'PERA VALLEJOS');
10
11 INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(6, 'GRB', 'Ian', 'RICHARDS');
12
13 INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(7, 'ESP', 'Sergio', 'SOLANA SOR');
14
15 INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(8, 'HKG', 'Man Chun', 'YEUNG');
16
```

- Log:** A table showing the execution log for the SQL command. The log entries are:

Time	Status	Command	Exec	Fetch	Rows	Message	SQL/Command
16:10:58	STARTED					Executing for: 'SQLite' [SQLite]	
16:10:58	SUCCESS	INSERT	0.070		1OK		INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName)
16:10:59	SUCCESS	INSERT	0.021		1OK		INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName)
16:10:59	SUCCESS	INSERT	0.020		1OK		INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName)
16:10:59	SUCCESS	INSERT	0.020		1OK		INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName)
16:10:59	SUCCESS	INSERT	0.020		1OK		INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName)
16:10:59	SUCCESS	INSERT	0.021		1OK		INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName)
16:10:59	SUCCESS	INSERT	0.019		1OK		INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName)
16:10:59	SUCCESS	INSERT	0.021		1OK		INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName)
16:10:59	FINISHED		0.212		0	8 ✓ Success: 8	

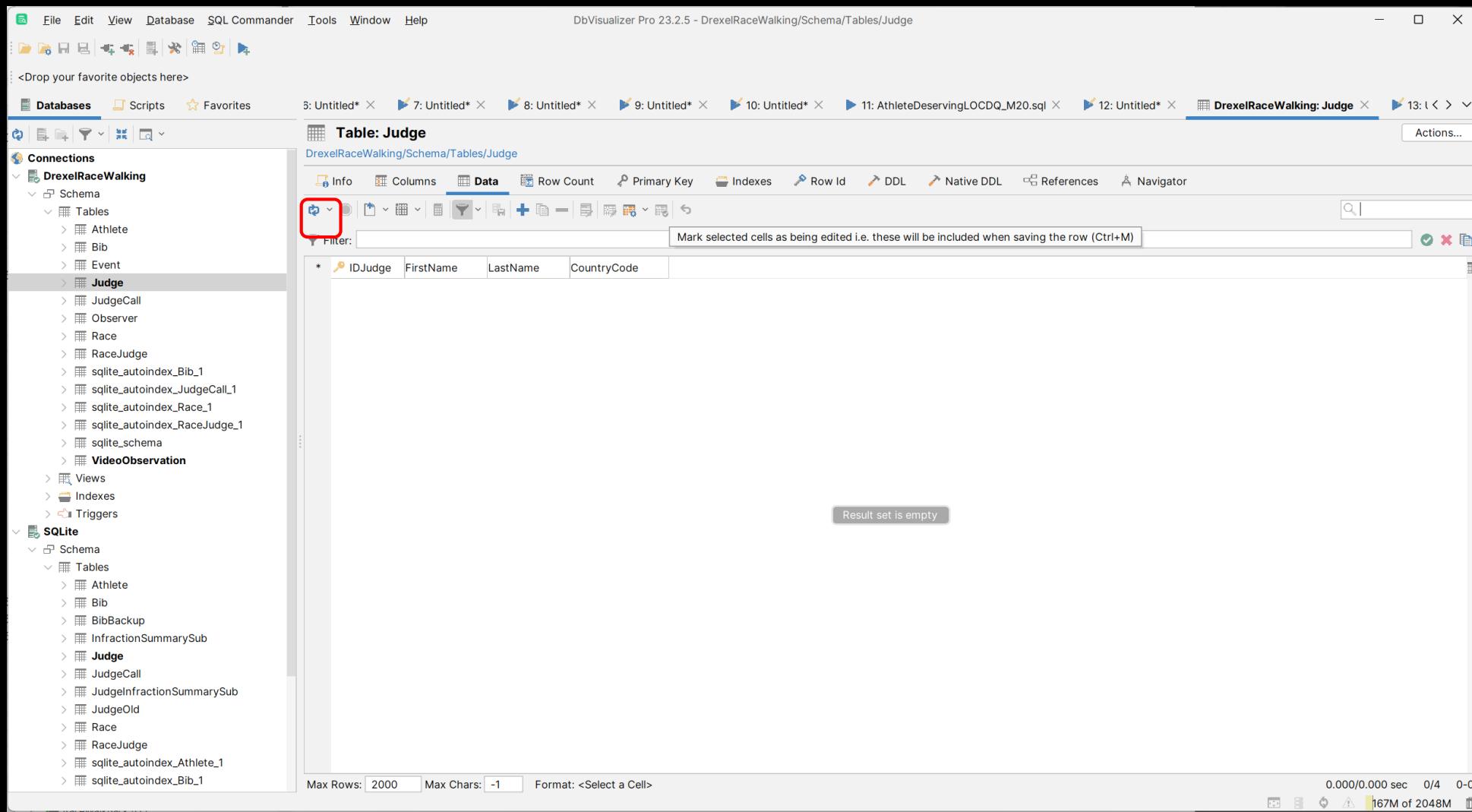
# Inserting Data – Via Code

Why is our data not there?

The screenshot shows the DbVisualizer Pro interface version 23.2.5. The title bar reads "DbVisualizer Pro 23.2.5 - DrexelRaceWalking/Schema/Tables/Judge". The left sidebar displays the database structure under "Connections" for "DrexelRaceWalking". The "Tables" section lists Athlete, Bib, Event, Judge, JudgeCall, Observer, Race, RaceJudge, and VideoObservation. The "SQLite" section also lists Athlete, Bib, BibBackup, InfractionSummarySub, Judge, JudgeCall, JudgeInfractionSummarySub, JudgeOld, Race, RaceJudge, and two autoindex tables. The main pane shows the "Table: Judge" with columns IDJudge, FirstName, LastName, and CountryCode. A red box highlights the entire data grid area, which is empty. A tooltip "Result set is empty" appears at the bottom right of the grid. The status bar at the bottom shows "Max Rows: 2000 Max Chars: -1 Format: <Select a Cell>".

# Inserting Data – Via Code

Hit the data *Refresh* button:



The screenshot shows the DbVisualizer Pro interface. The title bar reads "DbVisualizer Pro 23.2.5 - DrexelRaceWalking/Schema/Tables/Judge". The left sidebar displays the database structure under "Connections" for "DrexelRaceWalking" and "SQLite". The "Tables" section under "Judge" is expanded, showing various tables like Athlete, Bib, Event, Judge, etc. The main pane is titled "Table: Judge" and shows the "Data" tab selected. A table grid is present with columns: IDJudge, FirstName, LastName, and CountryCode. A tooltip above the grid states: "Mark selected cells as being edited i.e. these will be included when saving the row (Ctrl+M)". A message at the bottom right says "Result set is empty". The refresh button, located in the toolbar above the table grid, is highlighted with a red box.

# Inserting Data – Via Code

The data appears:

The screenshot shows the DbVisualizer Pro interface version 23.2.5. The main window title is "DbVisualizer Pro 23.2.5 - DrexelRaceWalking/Schema/Tables/Judge". The left sidebar displays the database connections and schema. Under "Connections", the "DrexelRaceWalking" connection is selected, showing its schema. The "Tables" section under "Judge" contains several tables: Athlete, Bib, Event, Judge, JudgeCall, Observer, Race, RaceJudge, sqlite\_autoindex\_Bib\_1, sqlite\_autoindex\_JudgeCall\_1, sqlite\_autoindex\_Race\_1, sqlite\_autoindex\_RaceJudge\_1, sqlite\_schema, VideoObservation, Views, Indexes, and Triggers. The "SQLite" connection also lists similar tables. The right panel shows the "Table: Judge" data view. The "Data" tab is selected, showing a grid of 8 rows of data. A red box highlights the first 8 rows of the table. The columns are labeled IDJudge, FirstName, LastName, and CountryCode. The data is as follows:

IDJudge	FirstName	LastName	CountryCode
1	Jean-Pierre	DAHM	FRA
2	José	DIAS	POR
3	Anne	FÖRBERG	FIN
4	Daniel	MICHAUD	CAN
5	Guillermo	PERA VALLEJOS	ARG
6	Ian	RICHARDS	GRB
7	Sergio	SOLANA SOR	ESP
8	Man Chun	YEUNG	HKG

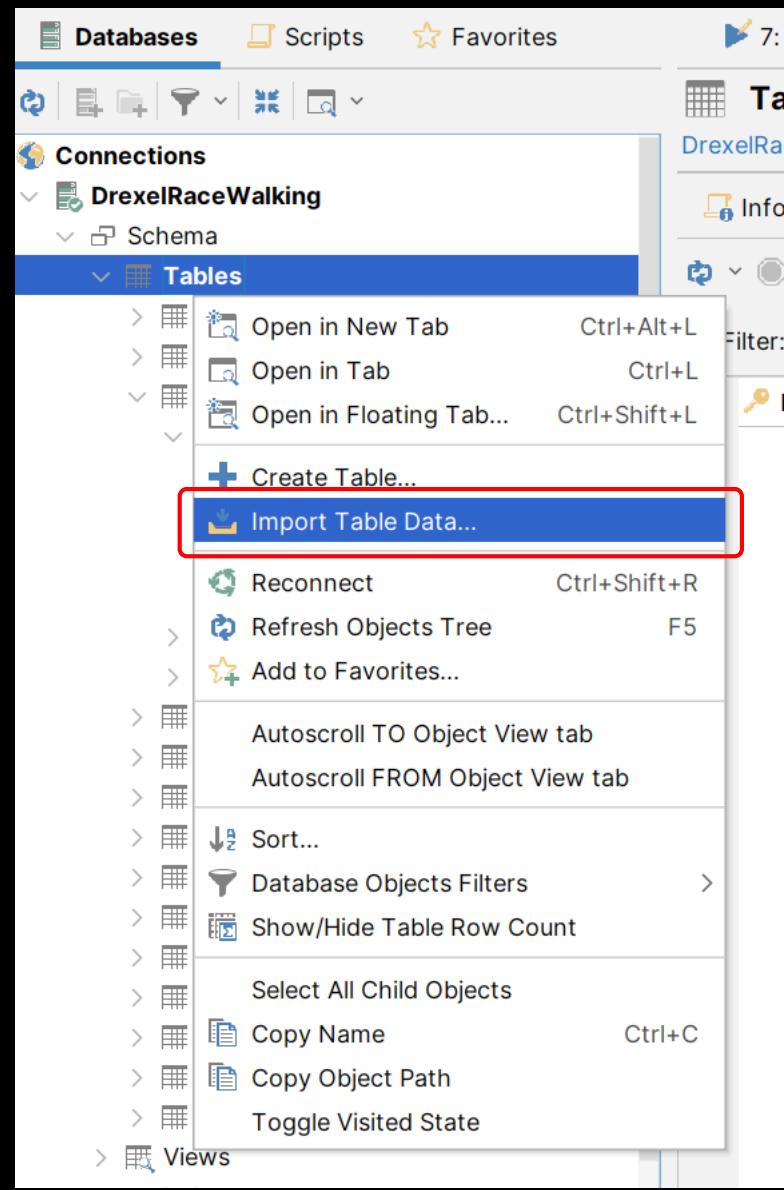
At the bottom of the interface, there are buttons for "Max Rows:", "Max Chars:", "Format:", and "Select a Cell". The status bar at the bottom right shows "0.000/0.000 sec" and "167M of 2048M".

# Create a Table from an Import

Shortcut the table creation process by creating it directly from an external data source, like a spreadsheet.

Right click on *Tables*

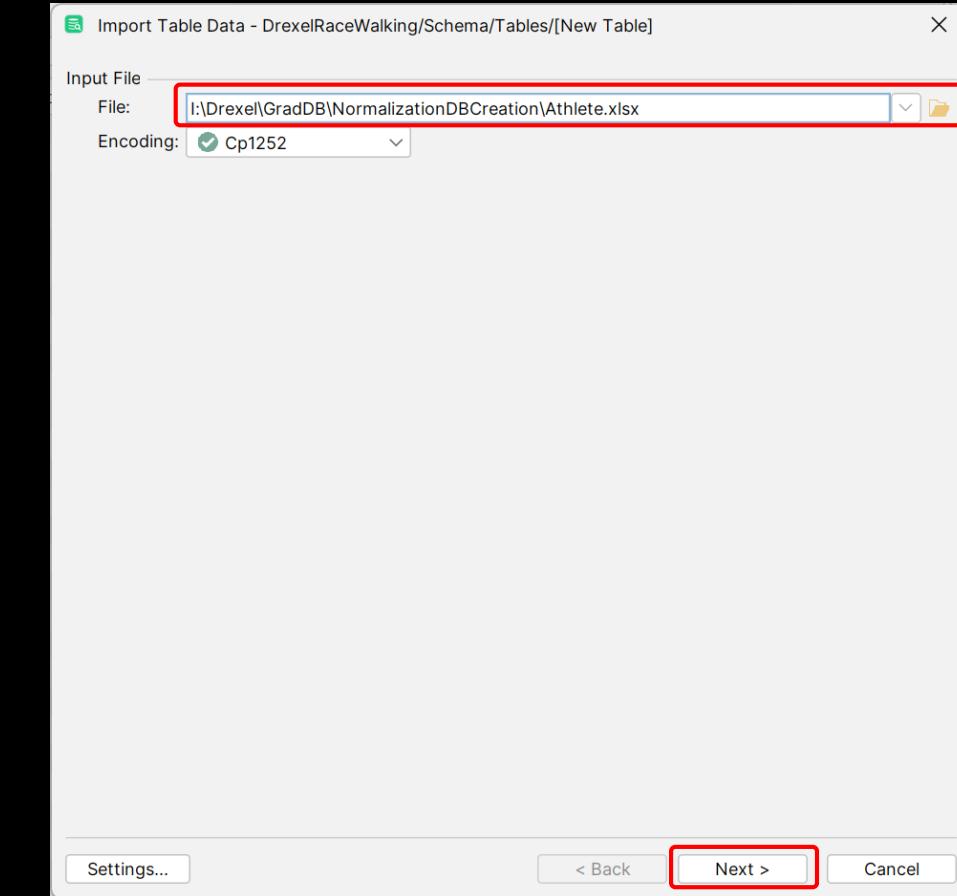
Click on *Import Table Data...* from the popup menu.



# Create a Table from an Import

Select a file to import:

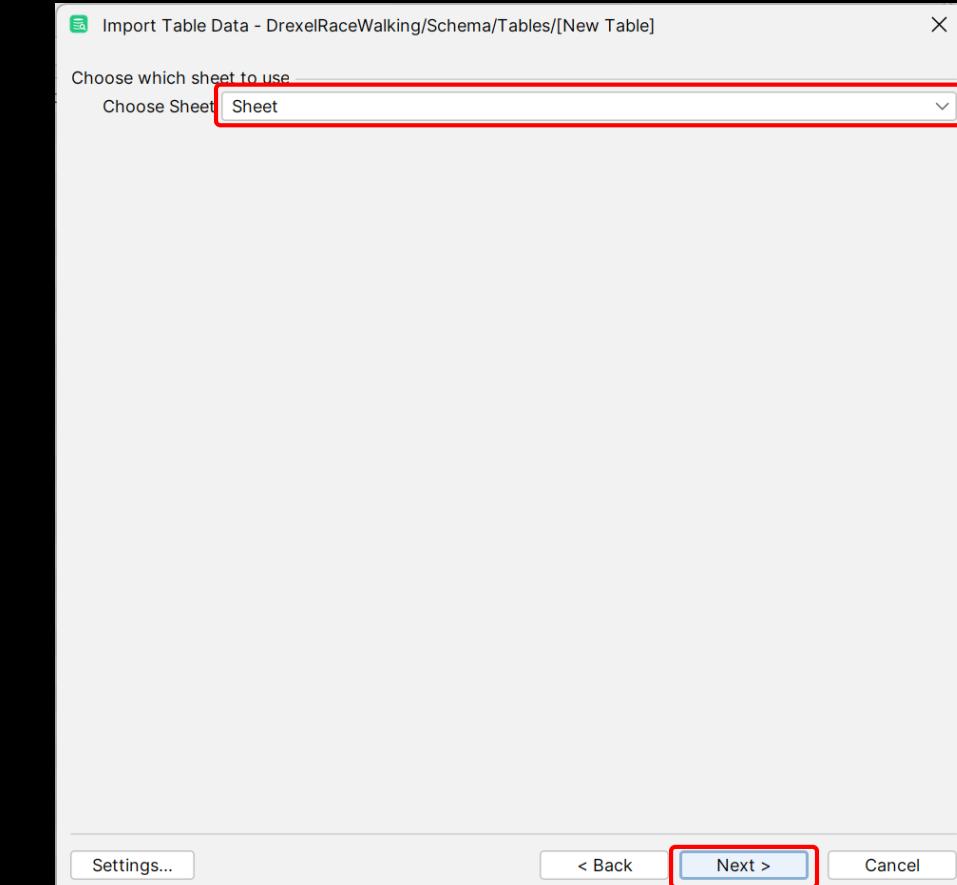
Then click the **Next** button:



# Create a Table from an Import

Since this is an Excel file, select the sheet the data is stored.

Then click the **Next** button:



# Create a Table from an Import

Assuming your data looks correct, click the **Next** button:

Import Table Data - DrexelRaceWalking/Schema/Tables/[New Table] X

Options

Row of Header:  Start Row of Data:  Skip Empty Row(s):

Data

**Grid**

IDAthlete	LastName	FirstName	CountryCo...	Gender
1	PÉREZ	María	ESP	F
2	GARCÍA LE...	Kimberly	PER	F
3	NTRISMPI...	Antigoni	GRE	F
4	LYRA	Viviane	BRA	F
5	MONTESI...	Cristina	ESP	F
6	INGA	Evelyn	PER	F
7	SONODA	Serena	JPN	F
8	CHOJECKA	Olga	POL	F
9	BONILLA	Magaly	ECU	F
10	ÄŽURDIAK...	Tereza	CZE	F
11	BAI	Xueying	CHN	F
12	ORTEGA	Alejandra	MEX	F
13	GONZÁLEZ	Raquel	ESP	F
14	FUCHISE	Masumi	JPN	F
15	CURIAZZI	Federica	ITA	F
16	MADARÁSZ	Viktória	HUN	F

Preview Rows:   Fit Column Widths

Settings... **Next >** **< Back** **Cancel**

# Create a Table from an Import

Assuming your data looks correct, click the **Next** button:

Import Table Data - DrexelRaceWalking/Schema/Tables/[New Table]

Data Formats

Date:	yyyy-MM-dd	2023-11-13
Time:	HH:mm:ss	18:41:16
Timestamp:	yyyy-MM-dd HH:mm:ss	2023-11-13 18:41:16
Number Separators: Grouping	,	Decimal
Boolean	True [e, yes, 1, on]	False [e, no, 0, off]
Null Value Text	(null)	

Data

Grid

IDAthlete	LastName	FirstName	CountryCo...	Gender
Decim...	String	String	String	String
1	PÉREZ	Maria	ESP	F
2	GARCÍA LE...	Kimberly	PER	F
3	NTRISMPI...	Antigoni	GRE	F
4	LYRA	Viviane	BRA	F
5	MONTESI...	Cristina	ESP	F
6	INGA	Evelyn	PER	F
7	SONODA	Serena	JPN	F
8	CHOJECKA	Olga	POL	F
9	BONILLA	Magaly	ECU	F
10	ÄŽURDIAK...	Tereza	CZE	F
11	BAI	Xueying	CHN	F

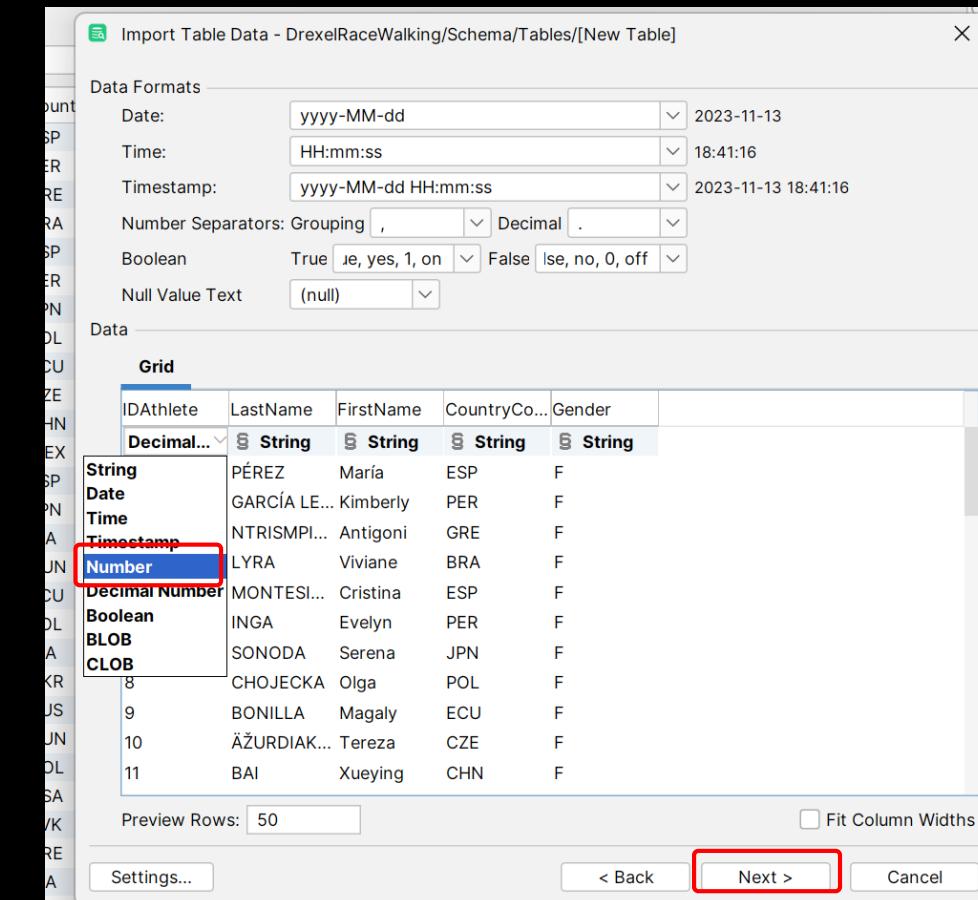
Preview Rows: 50  Fit Column Widths

Settings...  < Back  Next >  Cancel

# Create a Table from an Import

Change any data types DBVisualizer guesses wrong on:

Click the **Next** button:



# Create a Table from an Import

From here the process is the same as creating a table.

Give the table a name:

Indicate which fields are nullable:

Add any Primary and Foreign Keys:

Click the **Next** button:

Import Table Data - DrexelRaceWalking/Schema/Tables/[New Table]

Import Into  New Database Table

New Table Details

Database Connection: DrexelRaceWalking

Database:

Schema:

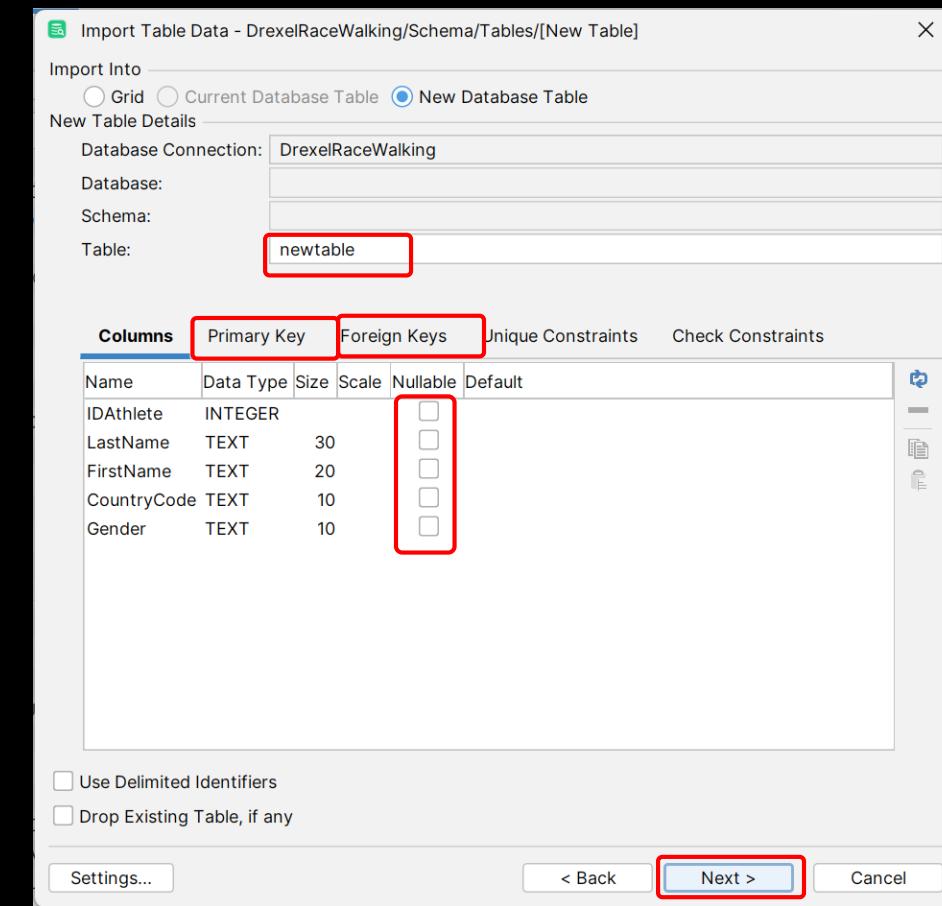
Table: newtable

Columns	Primary Key	Foreign Keys	Unique Constraints	Check Constraints	
Name	Data Type	Size	Scale	Nullable	Default
IDAthlete	INTEGER			<input type="checkbox"/>	
LastName	TEXT	30		<input type="checkbox"/>	
FirstName	TEXT	20		<input type="checkbox"/>	
CountryCode	TEXT	10		<input type="checkbox"/>	
Gender	TEXT	10		<input type="checkbox"/>	

Use Delimited Identifiers

Drop Existing Table, if any

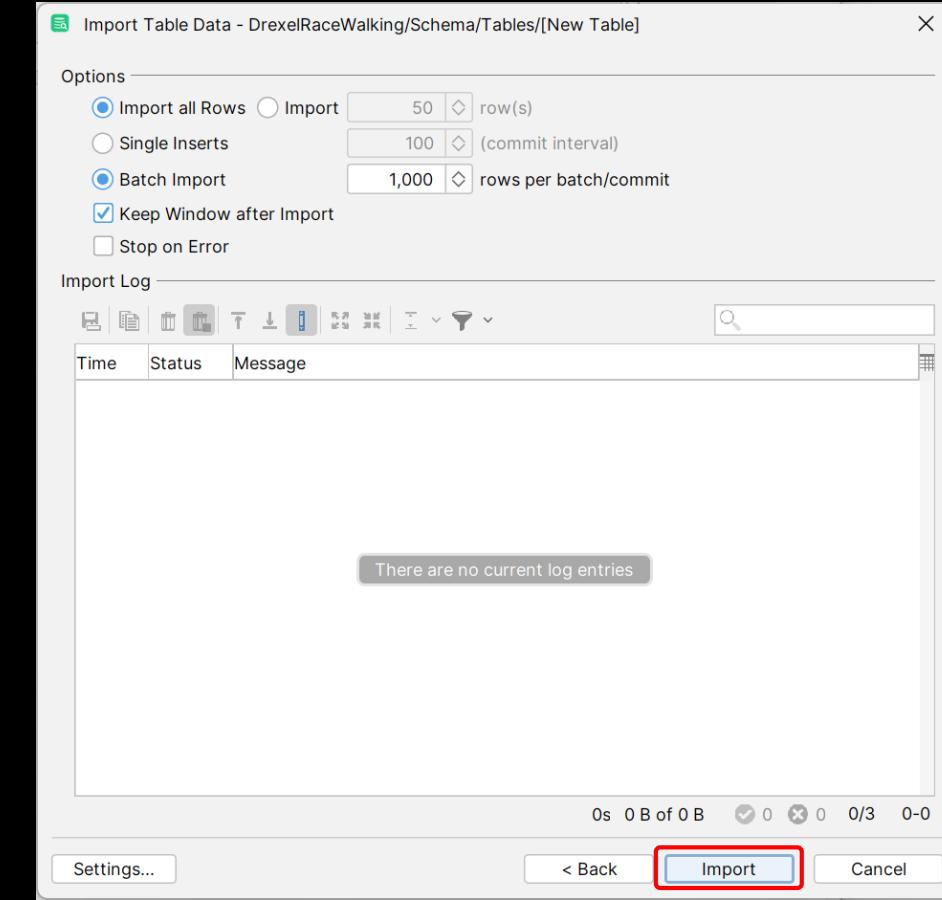
**Settings...** < Back **Next >** Cancel



# Create a Table from an Import

You are now ready to import the data:

Click the *Import* button:



# Create a Table from an Import

The status of the import is displayed:

Import Table Data - DrexelRaceWalking\Schema/Tables/[New Table]

Options

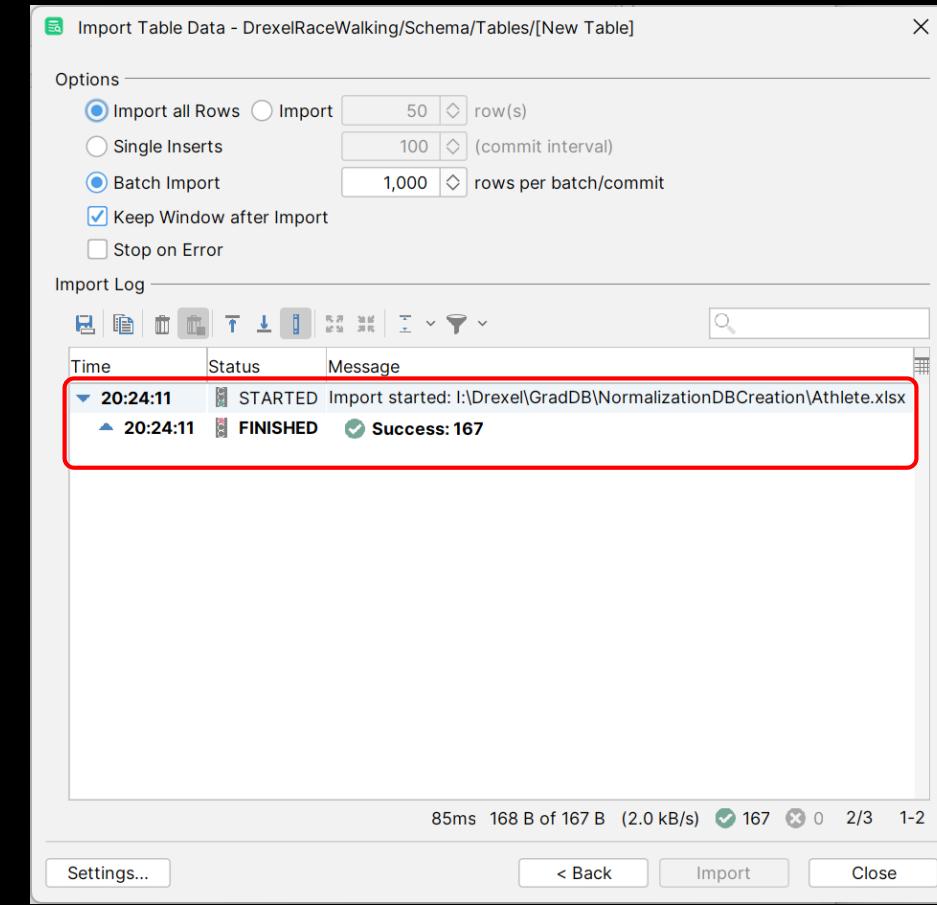
Import all Rows  Import 50 row(s)  
 Single Inserts 100 (commit interval)  
 Batch Import 1,000 rows per batch/commit  
 Keep Window after Import  
 Stop on Error

Import Log

Time	Status	Message
20:24:11	STARTED	Import started: I:\Drexel\GradDB\NormalizationDBCCreation\Athlete.xlsx
20:24:11	FINISHED	Success: 167

85ms 168 B of 167 B (2.0 kB/s) ✓ 167 ✘ 0 2/3 1-2

Settings... < Back Import Close



Time	Status	Message
20:24:11	STARTED	Import started: I:\Drexel\GradDB\NormalizationDBCCreation\Athlete.xlsx
20:24:11	FINISHED	Success: 167

# Viewing Data within DQVisualizer

Data in a single table can be displayed by double clicking on the table name in the *Schema* tree.

Observe viewing the *JudgeCall* table:

The screenshot shows the DbVisualizer Pro interface. The title bar reads "DbVisualizer Pro 23.2.5 - DrexelRaceWalking/Schem...". The menu bar includes File, Edit, View, Database, SQL Commander, Tools, Window, Help. The toolbar has icons for file operations like Open, Save, Print, and Database navigation. The left sidebar shows the "Connections" tree under "DrexelRaceWalking", with "Tables" expanded to show "Athlete", "Bib", "BibOriginal", "Event", "Infractions", "Judge", "JudgeCall" (which is highlighted with a red box), "NewJudgeCall", "Observer", "Race", "RaceJudge", "RandomName", "sqlite\_autoindex\_Bib\_1", "sqlite\_autoindex\_JudgeCall\_1", "sqlite\_autoindex\_RaceJudge\_1", "sqlite\_schema", "TempTable", "VideoObservation", "VideoObservationUpdate", "Views", "Indexes", and "Triggers". The right panel is titled "Table: JudgeCall" and shows the schema "DrexelRaceWalking/Schema/Tables/JudgeCall". It has tabs for Info, Columns, Data (which is selected), Row Count, Primary Key, and Indexes. A "Filter:" input field is present. The data grid displays 25 rows of data with the following columns: \* (row number), IDRace, IDJudge, Color, Infraction, TOD, and BibNumber. The data is as follows:

*	IDRace	IDJudge	Color	Infraction	TOD	BibNumber
1	1	1	1 Yellow	<	7:44:00 AM	102
2	3	2	Yellow	~	7:40:00 AM	101
3	1	3	Yellow	<	9:30:00 AM	102
4	3	3	Yellow	~	7:30:00 AM	100
5	3	3	Yellow	~	8:34:00 AM	101
6	1	5	Red	<	8:25:00 AM	102
7	1	5	Yellow	<	8:04:00 AM	102
8	1	6	Yellow	<	7:55:00 AM	102
9	3	7	Yellow	~	7:54:00 AM	101
10	3	8	Red	~	7:45:00 AM	100
11	3	8	Yellow	~	7:28:00 AM	100
12	3	7	Red	~	8:04:00 AM	101
13	3	1	Yellow	<	8:11:00 AM	103
14	3	2	Red	<	7:42:00 AM	103
15	3	2	Yellow	<	7:36:00 AM	103
16	3	3	Red	<	8:15:00 AM	103
17	3	3	Yellow	~	7:25:00 AM	103
18	3	7	Red	<	7:34:00 AM	103
19	3	7	Yellow	<	7:34:00 AM	103
20	3	8	Yellow	<	7:29:00 AM	103
21	3	8	Red	<	7:36:00 AM	103
22	3	2	Yellow	~	7:51:00 AM	105
23	3	1	Yellow	~	7:35:00 AM	105
24	1	1	Yellow	~	9:16:00 AM	106
25	3	1	Yellow	~	8:40:00 AM	106

At the bottom, there are buttons for Max Rows: 2000, Max Chars: -1, Number: Unformatted, and performance metrics: 0.001/0.002 sec, 638/6, 1/1, and 137M of 2048M.

# Viewing Data within DQVisualizer

Sort the data on a single field by clicking on the column header.

Observe the table sorted by *IDJudge*:

The screenshot shows the DbVisualizer Pro interface with the 'DrexelRaceWalking' database selected. The 'Tables' node under 'DrexelRaceWalking' is expanded, and the 'JudgeCall' table is selected. The 'Data' tab is active, displaying the contents of the JudgeCall table. The first column, 'IDRace', is highlighted with a red box. The second column, 'IDJudge', has an upward-pointing arrow icon above it, indicating it is the current sort key. The data in the table is sorted by 'IDJudge'. The table has columns: IDRace, IDJudge, Color, Infraction, TOD, BibNumber. The data rows show various race IDs, judge IDs, colors (Yellow or Red), infraction status (less than, greater than, or unsorted), times, and bib numbers.

*	IDRace	IDJudge	Color	Infraction	TOD	BibNumber
1	1	1	1 Yellow	<	7:44:00 AM	102
2	3	1	1 Yellow	<	8:11:00 AM	103
3	3	1	1 Yellow	~	7:35:00 AM	105
4	1	1	1 Yellow	~	9:16:00 AM	106
5	3	1	1 Yellow	~	8:40:00 AM	106
6	3	1	1 Yellow	~	8:05:00 AM	108
7	3	1	1 Yellow	~	8:20:00 AM	111
8	3	1	1 Yellow	<	7:42:00 AM	112
9	3	1	1 Yellow	~	7:54:00 AM	123
10	3	1	1 Red	~	7:59:00 AM	124
11	3	1	1 Yellow	~	7:44:00 AM	124
12	3	1	1 Yellow	~	7:57:00 AM	130
13	3	1	1 Yellow	<	7:30:00 AM	139
14	1	1	1 Yellow	~	8:48:00 AM	140
15	1	1	1 Red	<	8:04:00 AM	142
16	1	1	1 Yellow	<	7:35:00 AM	142
17	3	1	1 Yellow	~	8:05:00 AM	149
18	1	1	1 Yellow	~	9:21:00 AM	159
19	3	1	1 Yellow	<	7:34:00 AM	155
20	3	1	1 Red	<	7:49:00 AM	155
21	3	1	1 Yellow	~	7:38:00 AM	157
22	3	1	1 Yellow	<	8:22:00 AM	163
23	3	1	1 Red	~	8:31:00 AM	164
24	3	1	1 Yellow	~	7:33:00 AM	164
25	1	1	1 Yellow	<	9:09:00 AM	167

# Viewing Data within DQVisualizer

Sort the data on more than a single field by holding down the *ctrl* key while clicking on the column headers (in order) of the fields to sort.

The order of the sort is shown with small superscript numbers next to the column names.

Observe the table sorted by *IDJudge*, *IDRace*, *Color* and then *Infraction*:

The screenshot shows the DbVisualizer Pro interface with the 'DrexelRaceWalking' database selected. The 'JudgeCall' table is open in the 'Data' tab. The column headers are highlighted with a red box, showing superscript numbers indicating the sort order: IDRace^2, IDJudge^1, Color^3, and Infraction^4. The data in the table is sorted according to these criteria. The first few rows of the sorted data are as follows:

*	IDRace	IDJudge	Color	Infraction	TOD	BibNumber
1	1	1 Red	<	8:04:00 AM		142
2	1	1 Red	<	9:33:00 AM		181
3	1	1 Red	<	8:20:00 AM		183
4	1	1 Red	<	8:37:00 AM		329
5	1	1 Red	<	8:42:00 AM		326
6	1	1 Red	<	8:31:00 AM		351
7	1	1 Red	<	8:24:00 AM		372
8	1	1 Red	<	8:00:00 AM		378
9	1	1 Red	~	7:31:00 AM		312
10	1	1 Red	~	7:59:00 AM		368
11	1	1 Yellow	<	7:44:00 AM		102
12	1	1 Yellow	<	7:35:00 AM		142
13	1	1 Yellow	<	9:09:00 AM		167
14	1	1 Yellow	<	9:11:00 AM		181
15	1	1 Yellow	<	7:15:00 AM		172
16	1	1 Yellow	<	7:36:00 AM		183
17	1	1 Yellow	<	7:38:00 AM		304
18	1	1 Yellow	<	8:28:00 AM		329
19	1	1 Yellow	<	7:13:00 AM		326
20	1	1 Yellow	<	9:14:00 AM		307
21	1	1 Yellow	<	9:29:00 AM		309
22	1	1 Yellow	<	9:02:00 AM		379
23	1	1 Yellow	<	8:13:00 AM		351
24	1	1 Yellow	<	8:15:00 AM		372
25	1	1 Yellow	<	7:50:00 AM		378

# Viewing Data within DQVisualizer

Data can be filtered by adding conditions into the filter textbox.

Observe filtering for only `IDRace=2`:

Remember to click refresh after entering the filter data.

The screenshot shows the DbVisualizer Pro interface with the following details:

- Toolbar:** File, Edit, View, Database, SQL Commander, Tools, Window, Help, DbVisualizer Pro 23.2.5 - DrexelRaceWalking/Schem...
- Connections:** DrexelRaceWalking (selected), Schema, Tables, Athlete, Bib, BibOriginal, Event, Infractions, Judge, JudgeCall (selected), NewJudgeCall, Observer, Race, RaceJudge, RandomName, sqlite\_autoindex\_Bib\_1, sqlite\_autoindex\_JudgeCall\_1, sqlite\_autoindex\_RaceJudge\_1, sqlite\_schema, TempTable, VideoObservation, VideoObservationUpdate, Views, Indexes, Triggers, SQLite, Microsoft Access (UCanAccess).
- Table View:** Table: JudgeCall (DrexelRaceWalking/Schema/Tables/JudgeCall). The "Data" tab is selected. A filter bar at the top contains the text "Filter: WHERE IDRace=2".
- Data Grid:** A grid of 25 rows showing data from the JudgeCall table where IDRace is 2. The columns include IDRace (key), IDJudge (key), Color, Infraction, TOD, and BibNumber (key). The data shows various colors (Red, Yellow) and times (e.g., 11:43:00 AM, 11:27:00 AM, etc.) across multiple rows.
- Bottom Status:** Max Rows: 2000, Max Chars: -1, Number: Unformatted, 0.000/0.001 sec, 151/6, 1/1, 137M of 2048M.

# Viewing Data within DQVisualizer

Complex conditions are implemented with Boolean logic like AND / OR

Observe filtering for only `IDRace=2 AND Color = 'Red'`:

Note, remember to click refresh after entering the filter data.

The screenshot shows the DbVisualizer Pro interface with the 'DrexelRaceWalking' database selected. The 'JudgeCall' table is open in the main pane. A red box highlights the 'Filter:' input field at the top of the table view, which contains the SQL query `WHERE IDRace=2 AND Color = 'Red'`. The table data shows 25 rows of race data, with the first few rows matching the filter: IDRace 2 and Color 'Red'. The columns include IDRace, IDJudge, Color, Infraction, TOD, and BibNumber.

*	IDRace	IDJudge	Color	Infraction	TOD	BibNumber
1	2	1	Red	<	11:43:00 AM	34
2	2	1	Red	<	11:27:00 AM	34
3	2	1	Red	~	11:08:00 AM	30
4	2	1	Red	~	11:38:00 AM	36
5	2	1	Red	~	11:41:00 AM	38
6	2	2	Red	<	11:29:00 AM	36
7	2	2	Red	~	11:20:00 AM	3
8	2	2	Red	~	11:13:00 AM	31
9	2	2	Red	~	11:42:00 AM	32
10	2	2	Red	~	11:16:00 AM	33
11	2	3	Red	<	11:47:00 AM	30
12	2	3	Red	<	12:00:00 PM	3
13	2	3	Red	<	11:18:00 AM	34
14	2	3	Red	<	11:55:00 AM	35
15	2	3	Red	<	11:13:00 AM	36
16	2	3	Red	~	11:06:00 AM	30
17	2	3	Red	~	11:50:00 AM	30
18	2	3	Red	~	11:34:00 AM	31
19	2	3	Red	~	12:09:00 PM	36
20	2	4	Red	~	11:20:00 AM	30
21	2	4	Red	~	11:51:00 AM	30
22	2	4	Red	~	11:40:00 AM	3
23	2	4	Red	~	11:31:00 AM	33
24	2	4	Red	~	11:36:00 AM	35
25	2	4	Red	~	11:25:00 AM	36

# Viewing Data within DQVisualizer

SQLite is case sensitive when comparing data.

Not all databases are this way.

Observe filtering for `IDRace=2` and `Color = 'red'`:

The screenshot shows the DbVisualizer Pro interface version 23.2.5. The main window title is "DrexelRaceWalking/Schem...". The left sidebar shows a tree view of database connections, with "DrexelRaceWalking" selected. Under "Tables", the "JudgeCall" table is highlighted. The right panel displays the "Table: JudgeCall" data grid. A red box highlights the "Filter:" input field at the top of the grid, which contains the SQL query: "WHERE IDRace=2 AND Color = 'red'". Below the grid, a message box says "No rows match WHERE filter". The status bar at the bottom shows performance metrics: "0.001/0.000 sec 0/6 0-0".

# Autoincrement

Most databases have an option to create an integer field that is inserted automatically as a unique identifier.

Typically, the field is tagged with a qualifier like AUTO\_INCREMENT.

This is shown in the following example:

```
CREATE TABLE Car (
    IDCAR int NOT NULL AUTO_INCREMENT,
    Make varchar(255) NOT NULL,
    Model varchar(255),
    Year int,
    PRIMARY KEY (IDCAR)
);
```

SQLite doesn't support this.

# Autoincrement

SQLite does allow for auto increment, just by default.

To illicit the autoincrement behavior:

- set the auto increment field as the primary key
- this requires the field to not all nulls
- insert all the fields except the primary key

Observe the following table:

Observer		
IDObserver	FirstName	LastName
1	Jeff	Salvage
2	David	Harriman
3	JeffDavid	SalvageHarriman
4	Gary	Westerfield
5	JeffGary	SalvageWesterfield

IDObserver is the primary key.

To insert values and get the auto increment behavior, execute the following:

```
INSERT INTO OBSERVER (FirstName, LastName)  
VALUES ('Shmi', 'Skywalker')
```

Notice IDObserver is left off.

# Autoincrement

```
INSERT INTO OBSERVER (FirstName, LastName) VALUES ('Shmi', 'Skywalker')
```

Observer		
IDObserver	FirstName	LastName
1	Jeff	Salvage
2	David	Harriman
3	JeffDavid	SalvageHarriman
4	Gary	Westerfield
5	JeffGary	SalvageWesterfield



Observer		
IDObserver	FirstName	LastName
1	Jeff	Salvage
2	David	Harriman
3	JeffDavid	SalvageHarriman
4	Gary	Westerfield
5	JeffGary	SalvageWesterfield
6	Shmi	Skywalker

The 6 is inserted automatically and guaranteed to be unique.

# Inserting Data – Via Code

Assignment:

Install SQLite.

Install DBVisualizer Pro.

Create the entire database explained here.

This includes all the Primary and Foreign keys.

Save all the SQL statements required to create your table.

Load all the data for each table.

Manually enter the *Event* and *Race* tables.

Load from the given spreadsheets the data for the tables: *Athlete*, *Bib*, *Judge*, *JudgeCall*, *Observer*, *RaceJudge*, and *VideoObservation*.

Let's anonymize the data, use the following for Event and Race.

Event			
IDEvent	Event	City	Country
1	Boonta Eve Classic	Mos Espa	Tatooine

Race						
IDRace	IDEvent	RaceDate	StartTime	Distance	DistanceUnits	Gender
1		2023-08-24	7:00 AM	35	km	MF
2		2023-08-19	10:50 AM	20	km	M
3		2023-08-20	7:15 AM	20	km	F