SQL Programming

DDL

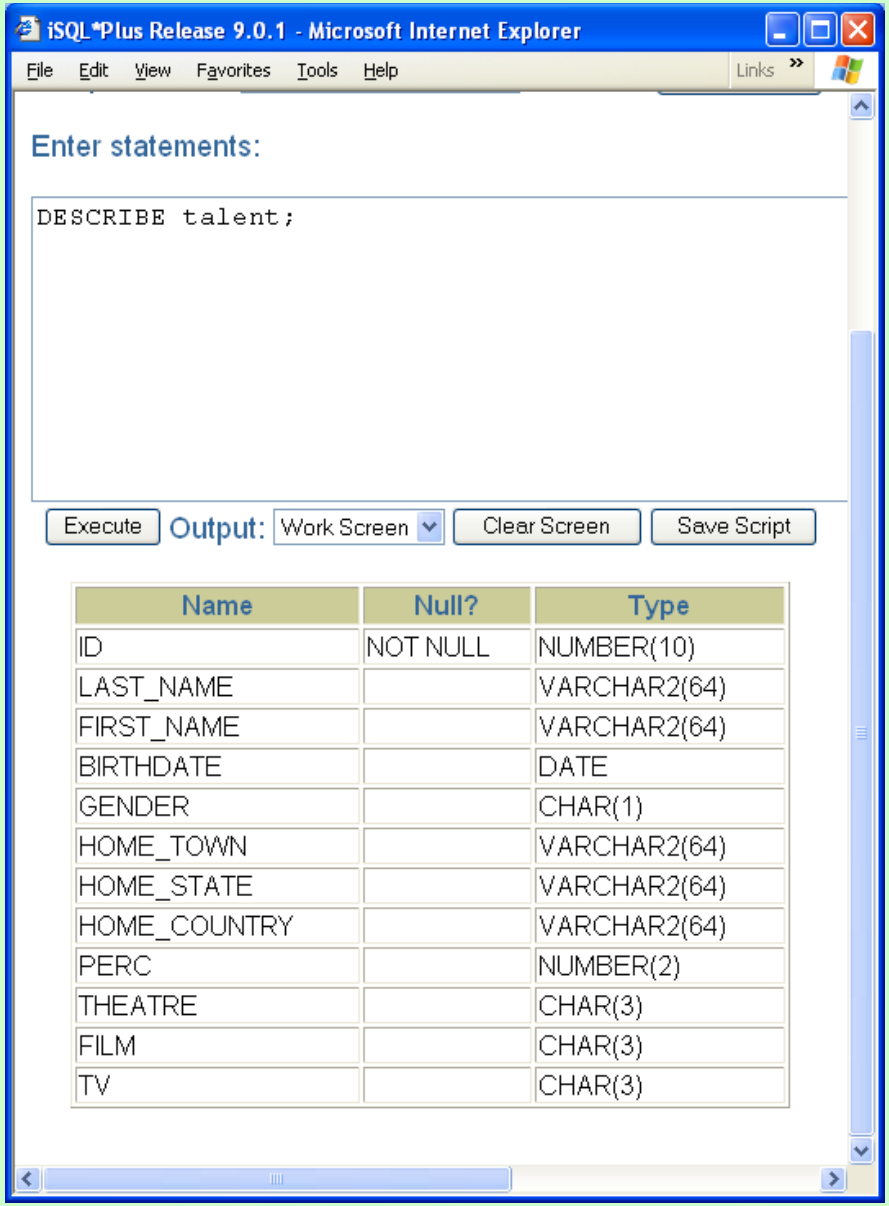
Module 17: DDL

Page A-1: Intro

In this ‘final’ module we examine the SQL statements that are used to create the database structures.

These statements are referred to as the data definition language (DDL) statements, because they are the statements that are used to DEFINE the DATA (Structures) in the database.

Page B-1 DESCRIBE



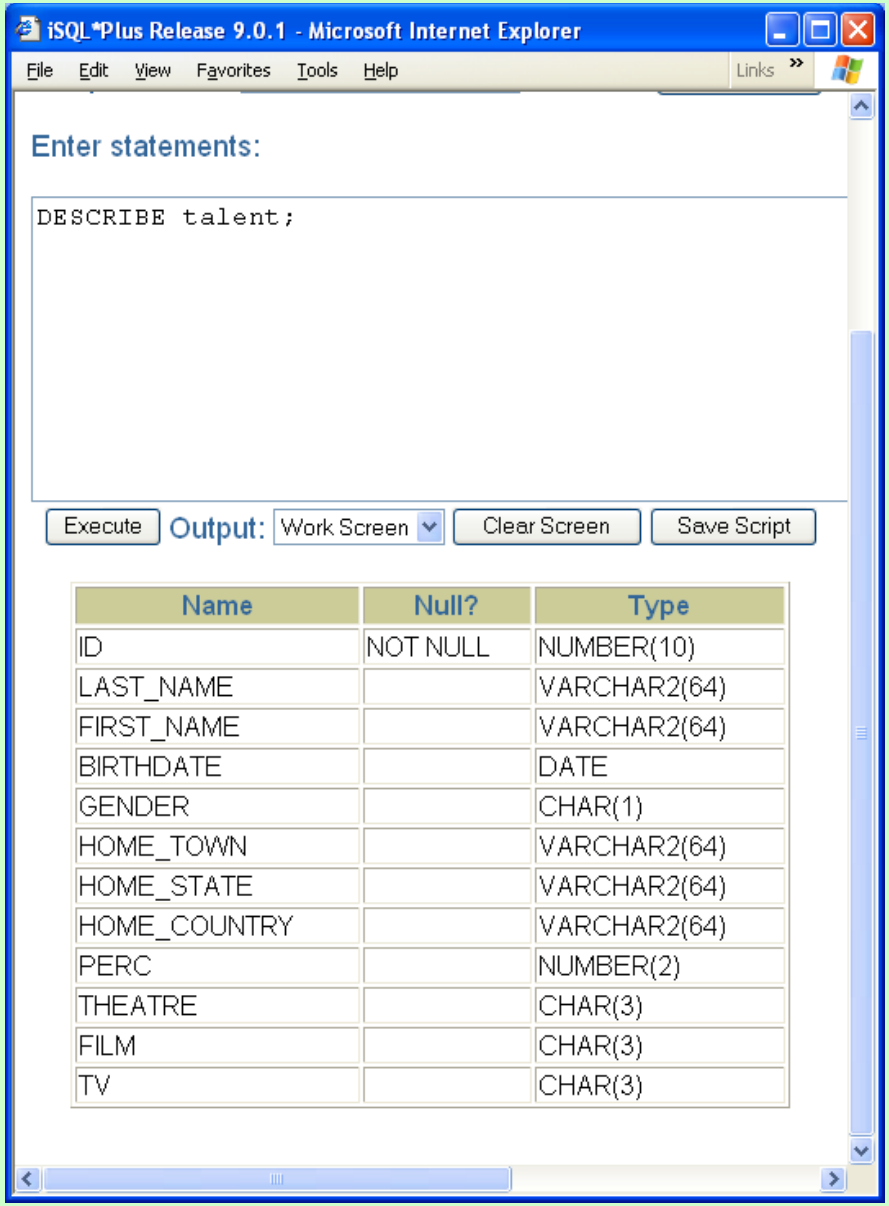
Name	Null?	Type
ID	NOT NULL	NUMBER(10)
LAST_NAME		VARCHAR2(64)
FIRST_NAME		VARCHAR2(64)
BIRTHDATE		DATE
GENDER		CHAR(1)
HOME_TOWN		VARCHAR2(64)
HOME_STATE		VARCHAR2(64)
HOME_COUNTRY		VARCHAR2(64)
PERC		NUMBER(2)
THEATRE		CHAR(3)
FILM		CHAR(3)
TV		CHAR(3)

We used the DECRIBE command before to examine the structure of a table.

Let's take another look at the talent table.

What are the structural elements of the talent table?

Page B-2 Table Structure



Name	Null?	Type
ID	NOT NULL	NUMBER(10)
LAST_NAME		VARCHAR2(64)
FIRST_NAME		VARCHAR2(64)
BIRTHDATE		DATE
GENDER		CHAR(1)
HOME_TOWN		VARCHAR2(64)
HOME_STATE		VARCHAR2(64)
HOME_COUNTRY		VARCHAR2(64)
PERC		NUMBER(2)
THEATRE		CHAR(3)
FILM		CHAR(3)
TV		CHAR(3)

Every table has a name.

Every table contains at least one column.

Every column has a name.

Every column is of a particular datatype and in this regard we often say that columns define a domain of values from which their members may be drawn.

The domain specifies the following:

Datatype: number, character, date

Size/format: eg. 30 character name field,
date in DD-Mon-YYYY format

Page B-3 CREATE TABLE

The CREATE command is used to create structures in the database.

And the syntax for the CREATE statement is something like this:

CREATE *structure-type structure-name*
[*details about that structure*]

Let's build a table named music.

What kind of structure are we building?

What kind of structure are we building?

What kind of structure are we building?

CREATE *structure-type structure-name*
[*details about that structure*]

A table

CREATE TABLE *structure-name*
[*details about that structure*]

So step one is to specify TABLE as our structure type.

What's the name for this structure?

CREATE *structure-type structure-name*
[*details about that structure*]

What kind of structure are we building?

A table

CREATE TABLE *structure-name*
[*details about that structure*]

So step one is to specify TABLE as our structure type.

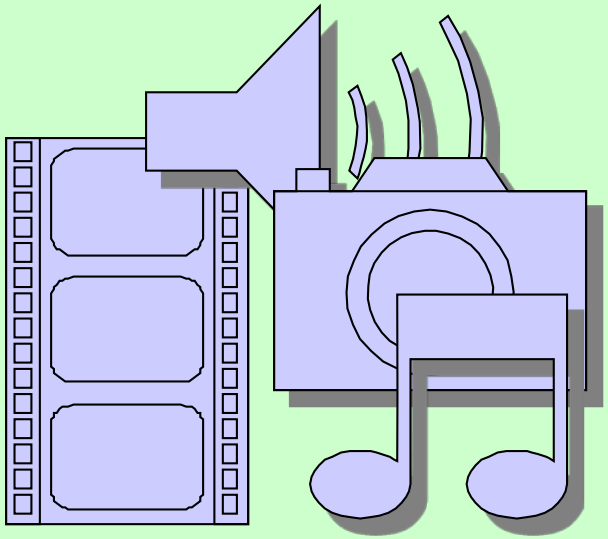
What's the name for this structure?

music

CREATE TABLE music
[*details about that structure*]

So the next step is to plug in the name of our table

Now all we have to do is 'flesh out' the details.



I'd like the music table to keep track of the music that I have at home in my CD collection.

This will be a simple table with only a few fields of information.

I want to know the name of the artist (or band), and the title of the album. As well as the media/format, eg. CD, cassette, MP3.

Oh yeah, I read somewhere that records in a database should have a key? Or an id number? Or something like that. So let's throw one of those in too.

Now this process of investigating my needs and coming up with a suitable table falls under the work heading of systems analysis (and design) or database analysis and design. You're only a programmer! Or more correctly, this is only a *programming* class. So we won't delve into the design aspects.

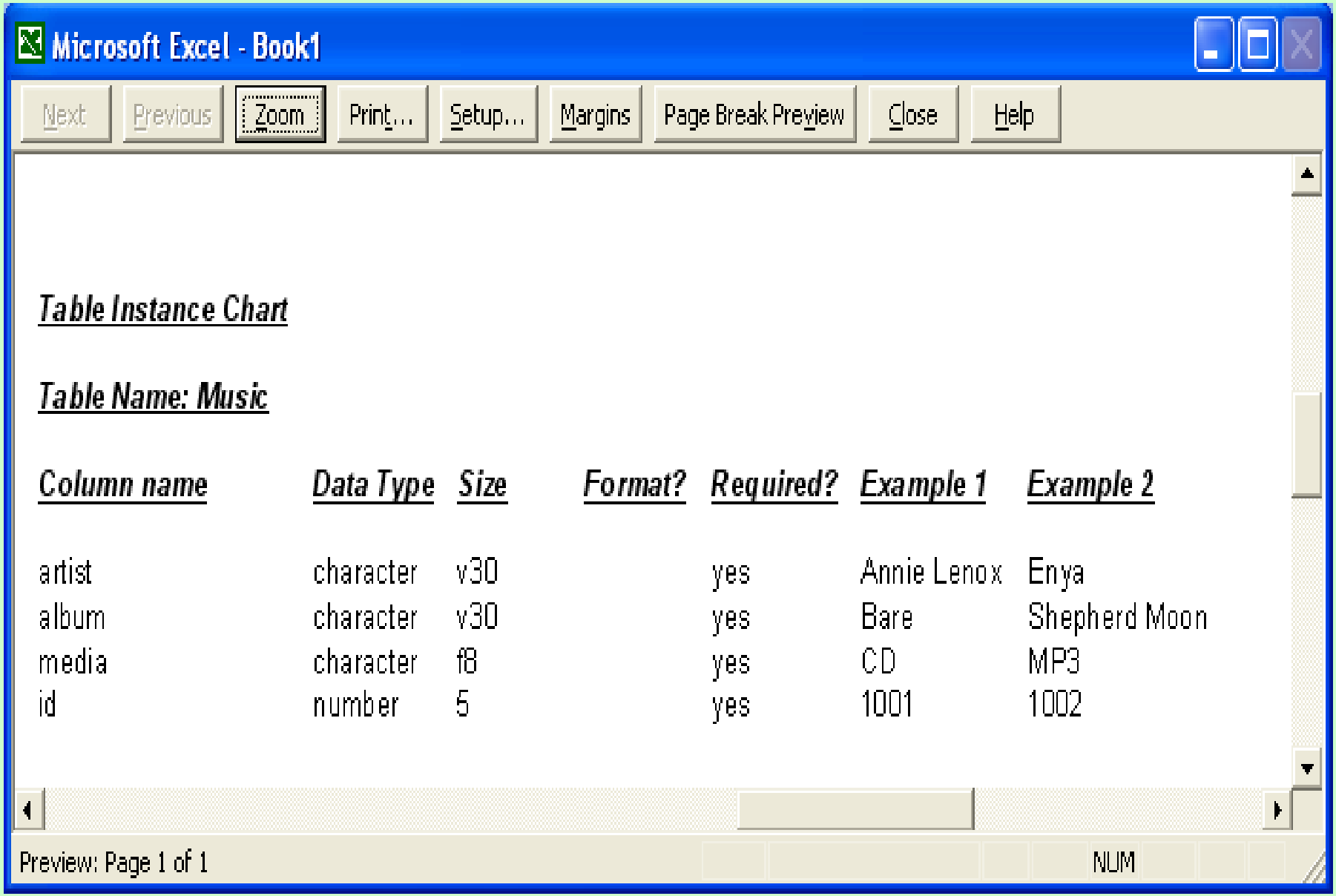
Although you won't develop the design, as a programmer you'll be expected to read the design and then implement all of the features specified therein.

Designs generally come in one of two formats:

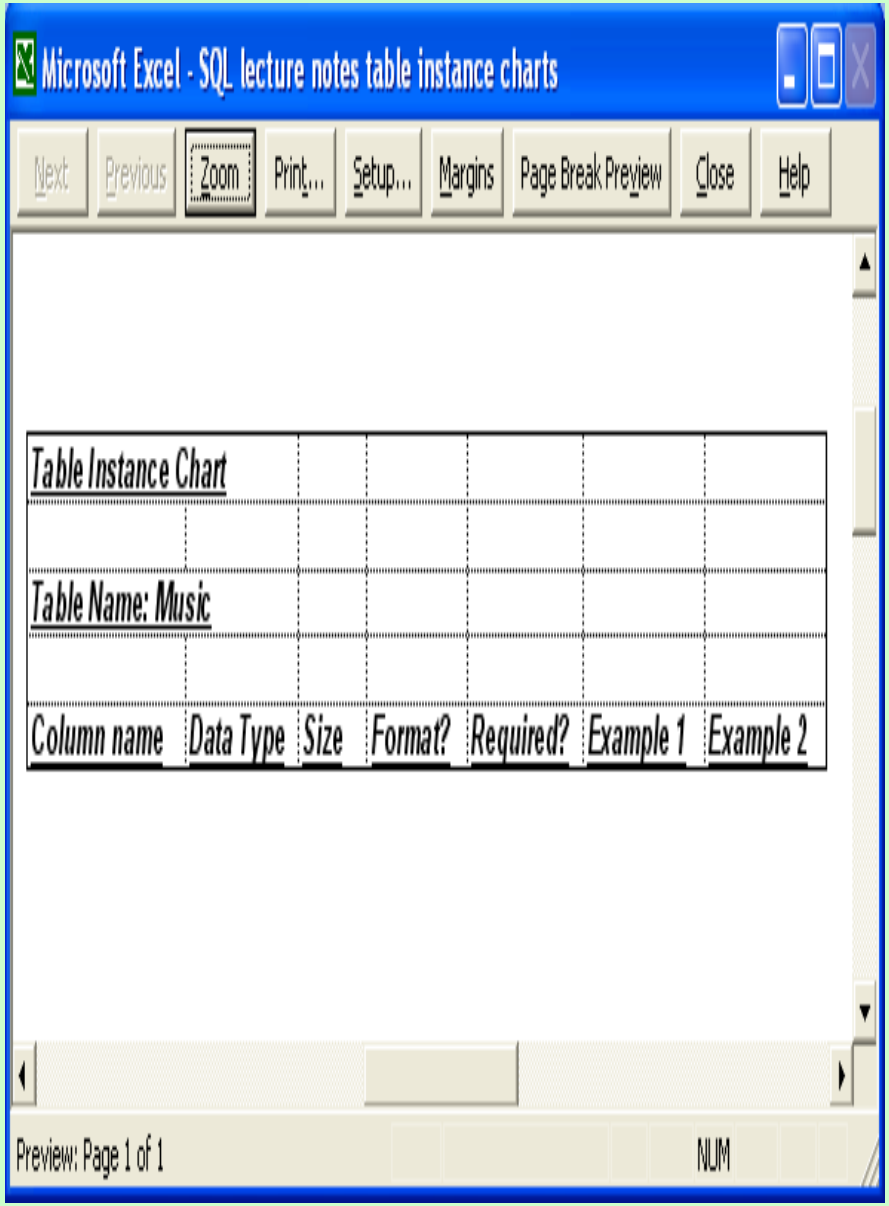
- Table Instance Charts
- Entity Relationship Diagrams

In this module we're only going to focus on Table Instance Charts

B-8 Table Instance Chart - Example



Page B-9 Table Instance Chart



The general format for a table instance chart allows the designer to specify these features about the table:

Table name

Column name
Data type
Size
Format
Required?
Samples

So now all you need to know as the programmer is how to map a table instance chart into the SQL CREATE statement.

Page C-1 Datatype: Character

Datatypes come in three main flavors: character, numeric, and date.

As to character datatype, they are defined in the database in one of two ways: either as a fixed length character field, or as a variable length character field.

Which of these two subtypes you choose is based primarily on performance (and space) considerations.

A fixed length 10 character field, will always use 10 characters, regardless of the size of the entry. A variable length 10 character field will allow no more than 10 characters in the entry, and it will readjust the amount of space it takes up, based on the size of the entry.

Fixed length character fields are defined with the CHAR datatype, followed by the size of the column in parentheses, eg.

```
name  CHAR(30),
city  CHAR(20)
```

Variable length character fields (in Oracle) are defined with the VARCHAR2 datatype, in a similar fashion, eg.

```
major  VARCHAR2(16),
street VARCHAR2(32)
```

Page C-3 Datatype: Number

Number fields are defined with the NUMBER datatype.

Integer values, or whole numbers, are defined simply be specifying the precision (number of digits) that you need for the value.

2-digit value	NUMBER(2)
5-digit value	NUMBER(5)
9-digit value	NUMBER(9)

Page C-4 Datatype: Number

Decimal values, require you to specify both the number of digits that are to be displayed, as well as how many positions to the right of the decimal point are to be permitted.

Here are a few examples of sample data, and the datatype specification needed to correctly store that value.

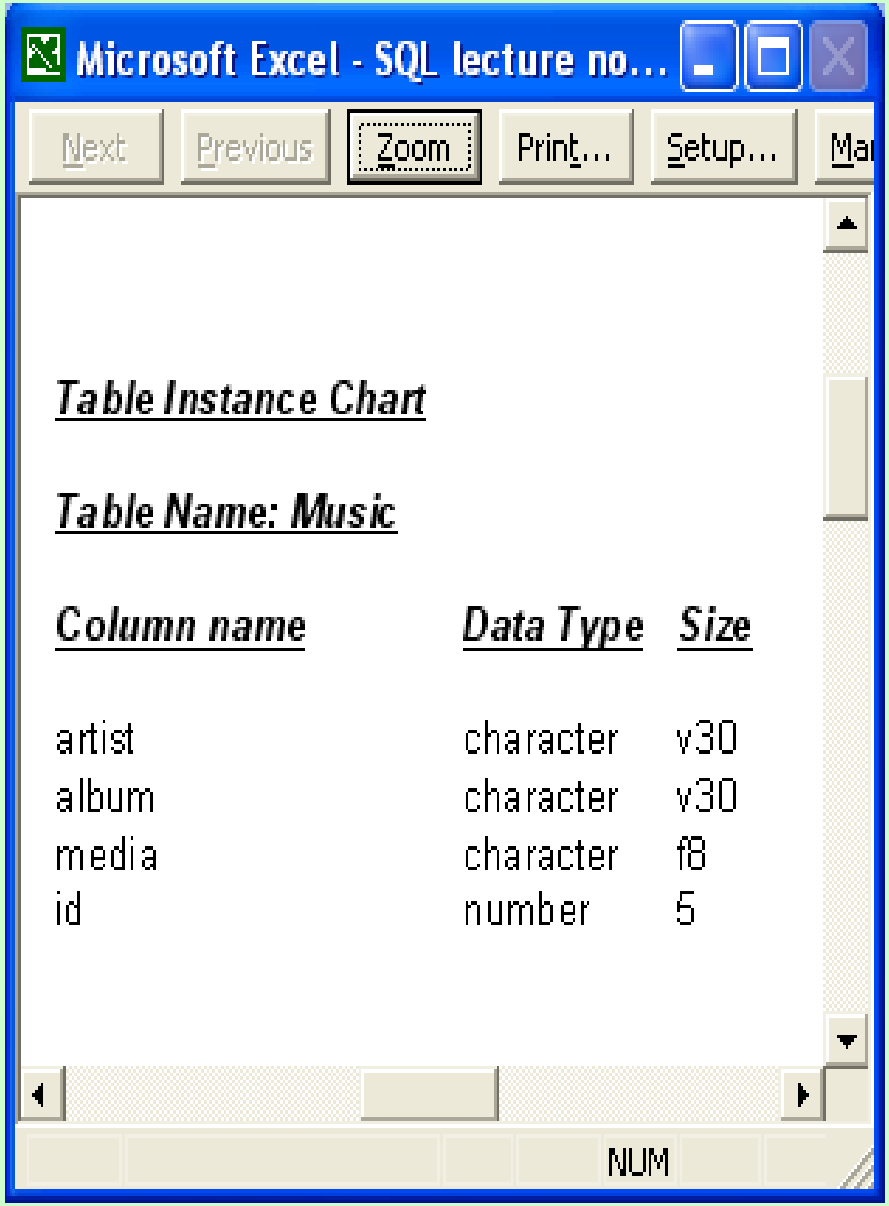
Value	Definition
30	Number (2)
30.1	NUMBER(3,1)
30.22	NUMBER(4,2)
30000.123	NUMBER(8,3)

Page C-5 Datatype: Date

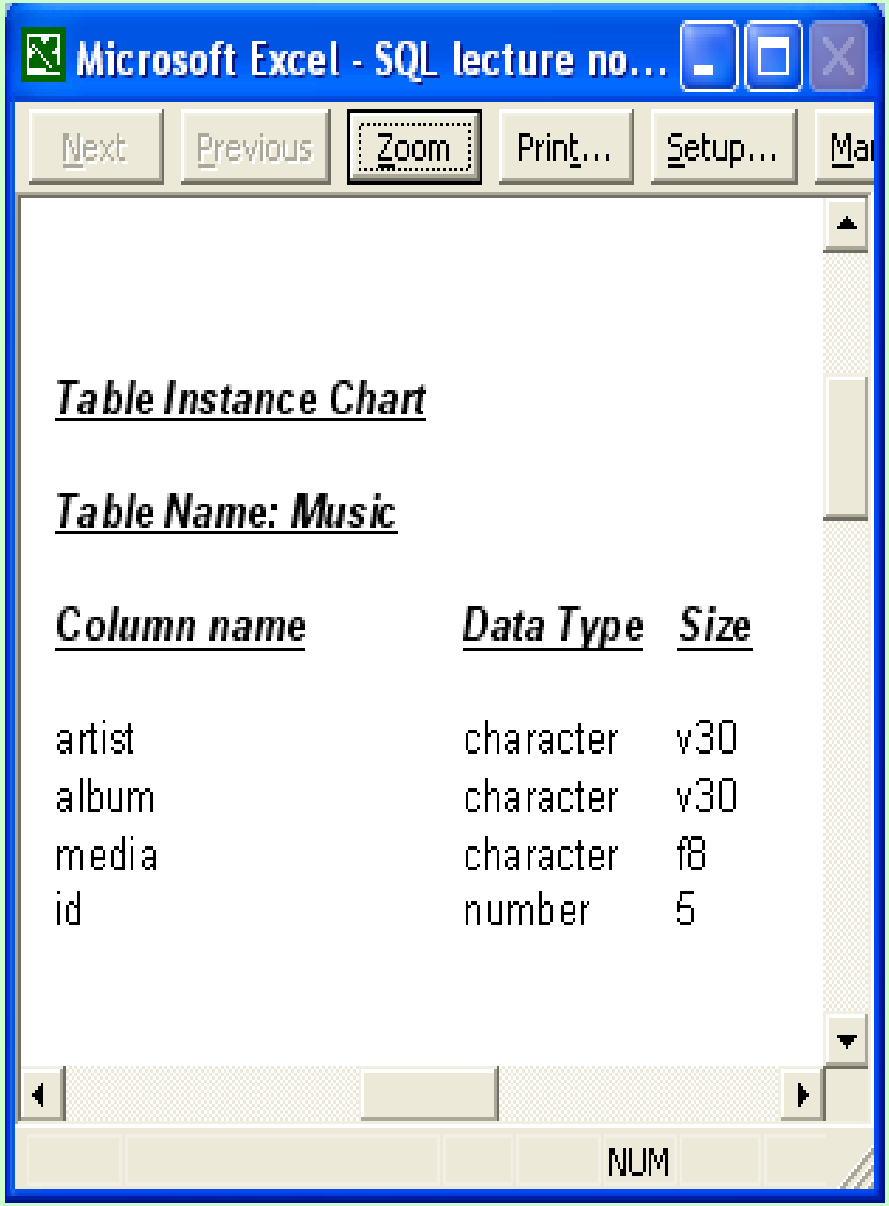
Date (and time) values are stored in the database using the DATE datatype.

For example,

```
birthdate          DATE,
application_date   DATE
```



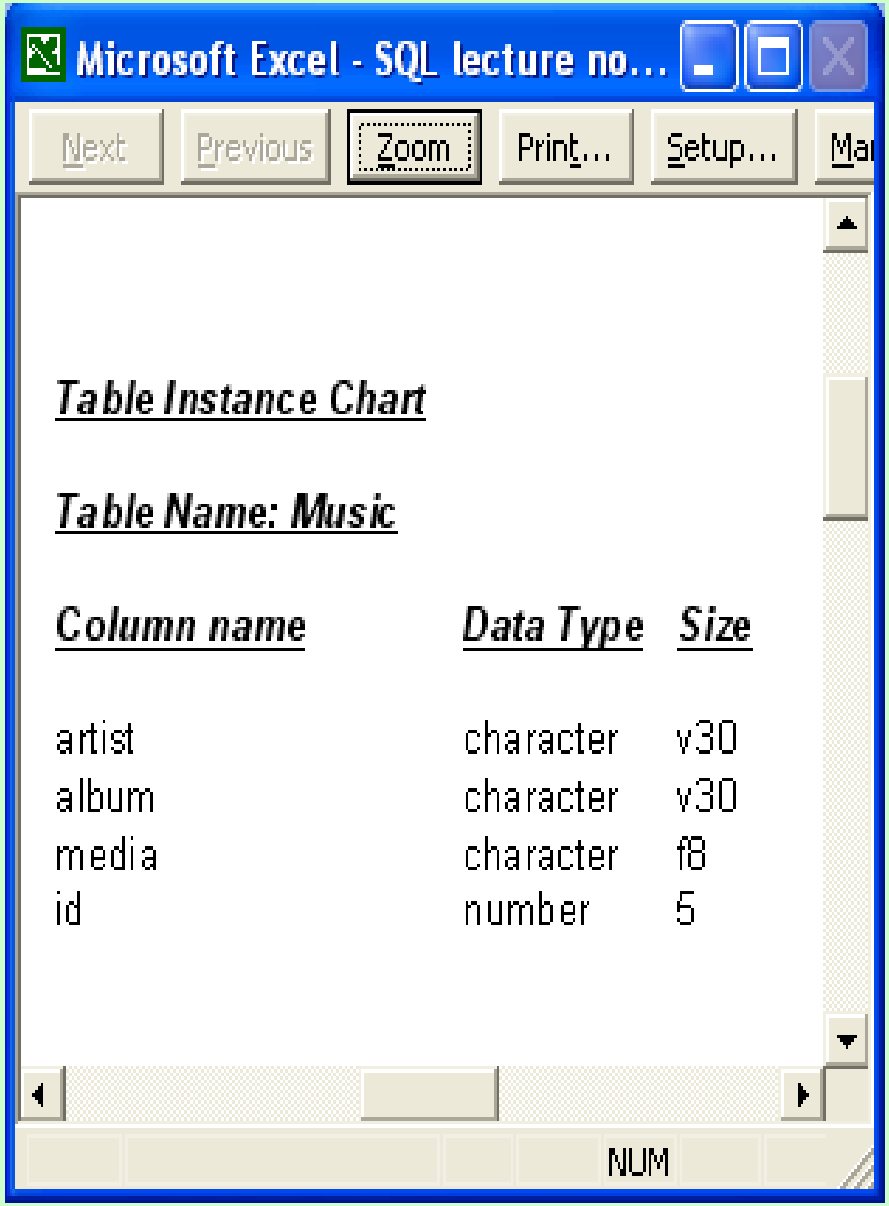
Let's return to our table instance chart and finish the program we re writing to create the music table structure.



We had gotten this far:

CREATE TABLE music
[*details about that structure*]

Now we need to provide the details for the table structure. Minimally we need to specify the name and datatype for each of the columns in the table.



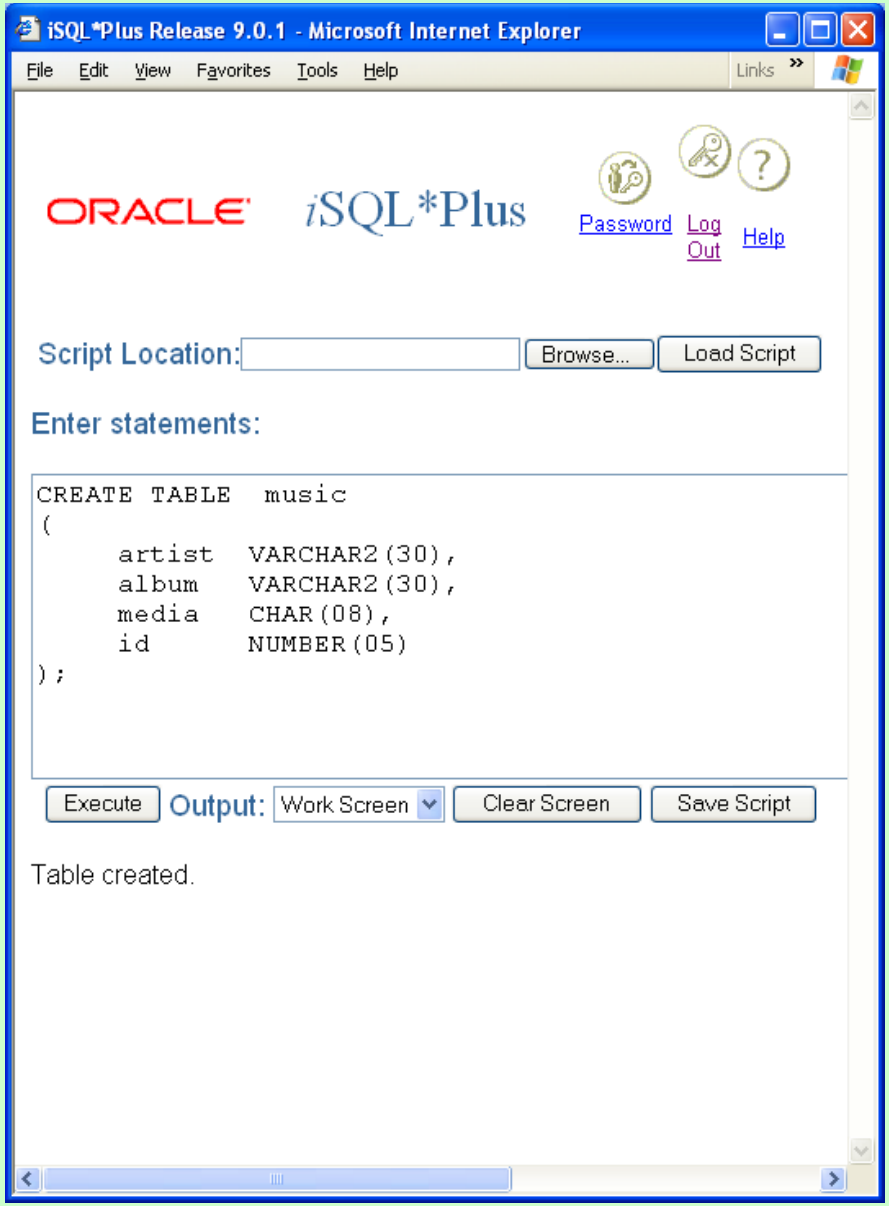
All of the column information immediately follows the table-name, and needs to be enclosed within parentheses.

So the syntax now looks something like this:

```
CREATE TABLE  table-name
(
        column-name1   datatype,
        column-name2   datatype
        ...
);
```

And we should come up with something like this for our music table:

```
CREATE TABLE  music
(
    artist    VARCHAR2(30),
    album     VARCHAR2(30),
    media     CHAR(08),
    id        NUMBER(05)
);
```



Notice the way I've used white space to highlight my code, and hopefully, make it more understandable.

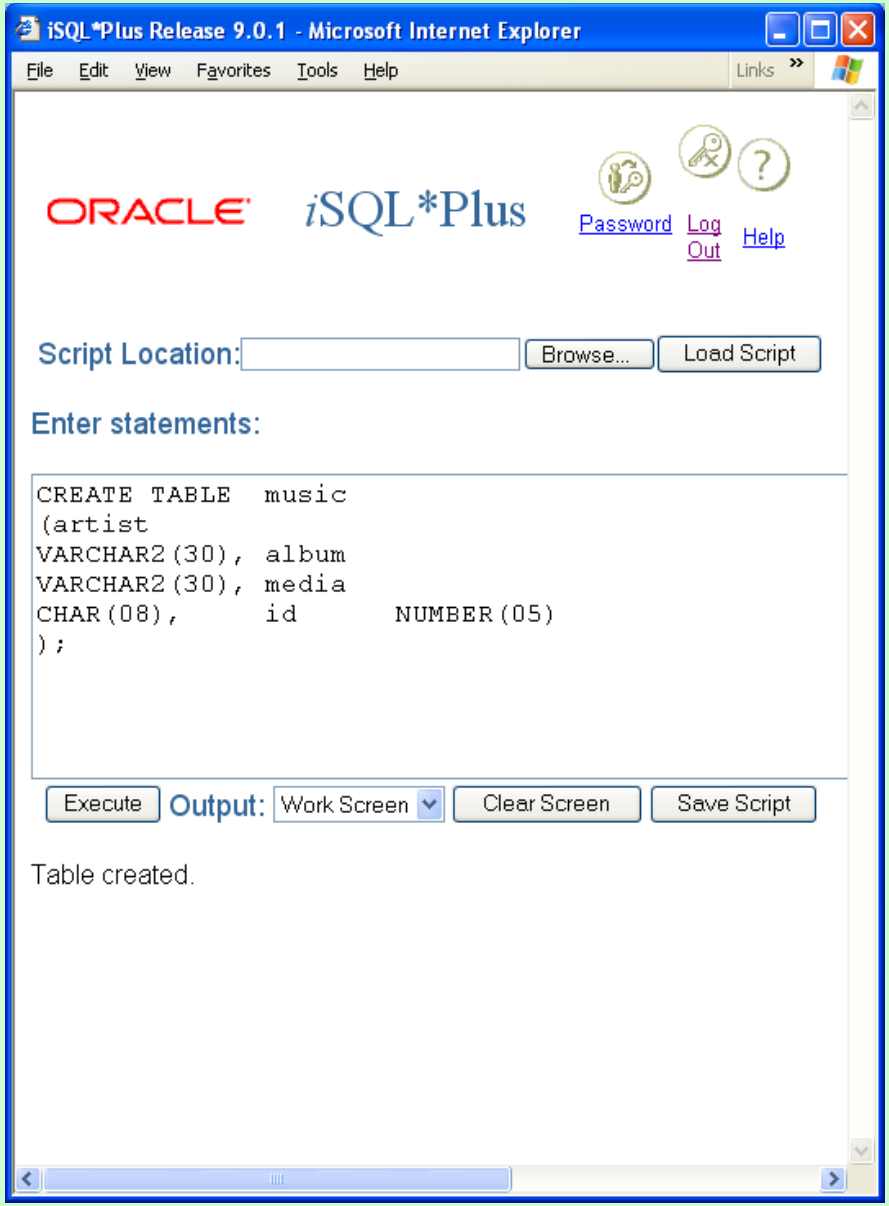
```
CREATE TABLE  music
(
    artist   VARCHAR2(30),
    album    VARCHAR2(30),
    media    CHAR(08),
    id       NUMBER(05)
);
```

CREATE TABLE stands on a line of its own.

All details are indented underneath that 'heading'.

Details are coded in a columnar fashion, ie. Column names appears to be in a column of their own, datatypes appear to be in a column of their own.

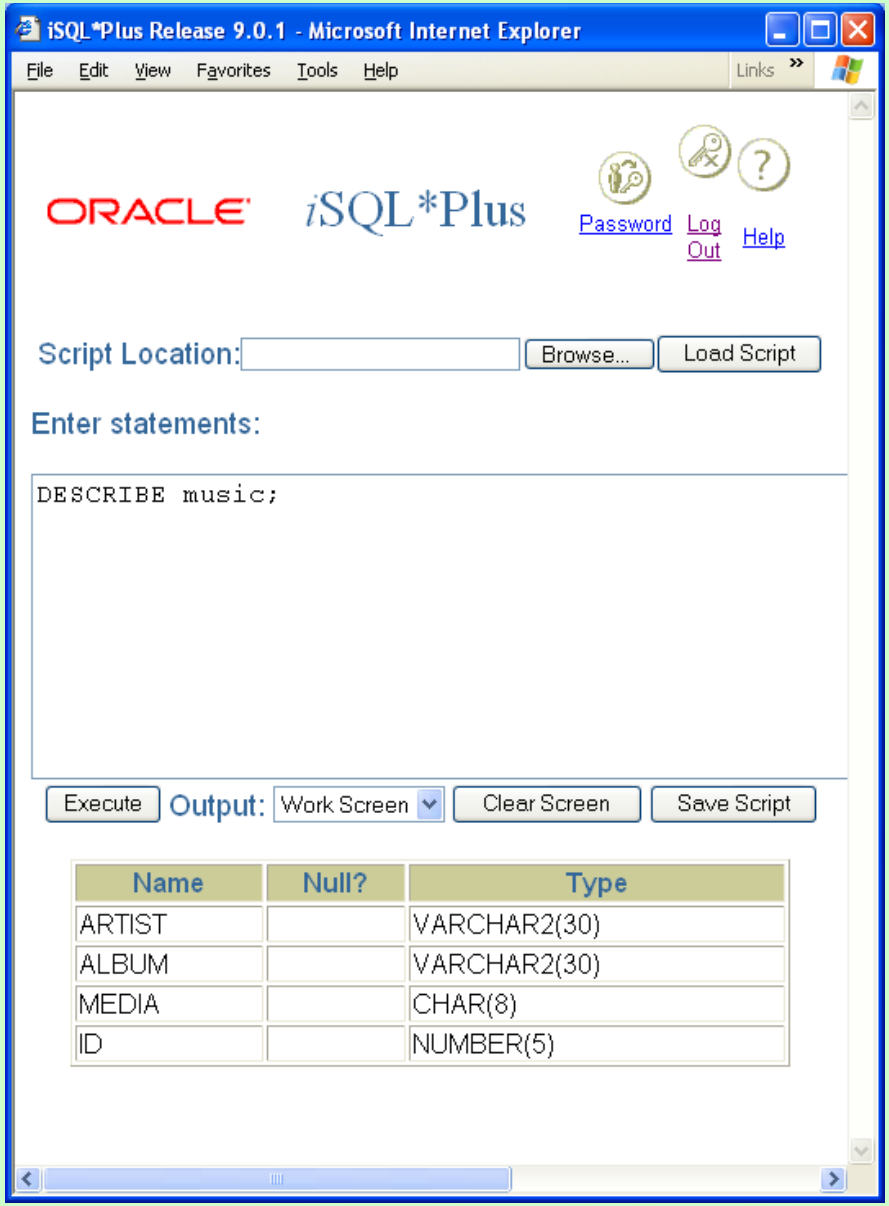
One columns worth of information at a time.



Remember, these conventions are for the reader, not SQL.

SQL would be happy with this code.

Page D-6 Verify



Use the describe command to check that the table was created as you had intended.

Page E-1 Virtual Tables

Now that you can build tables that can store data, let's look at building virtual tables.

What's a virtual table?

A virtual table is a 'fake' table. It doesn't store any data, it just behaves like a table. In this regard, it has the behavior of a table, but not the data.

Pretty cool huh?

You can actually build something in the database that won't store data, but acts like a table???

Who dreamed this up? And why the heck would you ever use it?

Give me a moment and we'll get there, but first,

A virtual table is stored in the database as a VIEW structure.

I realize I just said that a virtual table (View) doesn't store data, so what is it that's getting stored in the database?

A SQL program, specifically a SELECT statement.

Think of a view as a SELECT subquery. Subqueries don't store data do they? But they do behave like tables, don't they?

So back to the question we phrased a moment ago.

Why? What's the point of having a view?

Page E-3 Views

A VIEW can be set up to give different user groups their own *views* of the database.

Consider this scenario. The Personnel department stores all of the information about every employee in a single table. But not everyone in the company should have access to all of that information.

The phone directory office should be able to access name department and phone number, and nothing else. Line managers should be able to access information about their team, but nothing about other team members.

The two primary reasons for using views are to provide a level of security, and to simplify access.

I have a few friends and we share our movies between us.

I've got some old BETA tapes and LaserDiscs that none of these folks are interested in. They only borrow my DVDs.

They'll hook up to my database and check my collection to see if there's anything they want to watch. But every time they run a query they need to include this predicate:

```
WHERE  format <> 'DVD'
```

I want to create a view that'll simplify their using my database

Page E-5 CREATE VIEW syntax

Recall the general syntax description for the CREATE statement. It went something like this:

CREATE *structure-type structure-name*
[*details about that structure*]

When creating a view structure you specify the term VIEW, ie.

CREATE VIEW *structure-name*
[*details about that structure*]

Each view needs to be named, and the details for view structure is simply a SELECT statement.

To design a view I need a name, and a SELECT program.

CREATE VIEW *structure-name*
[*SELECT program*]

Step 1. Name the view

CREATE VIEW dvd AS

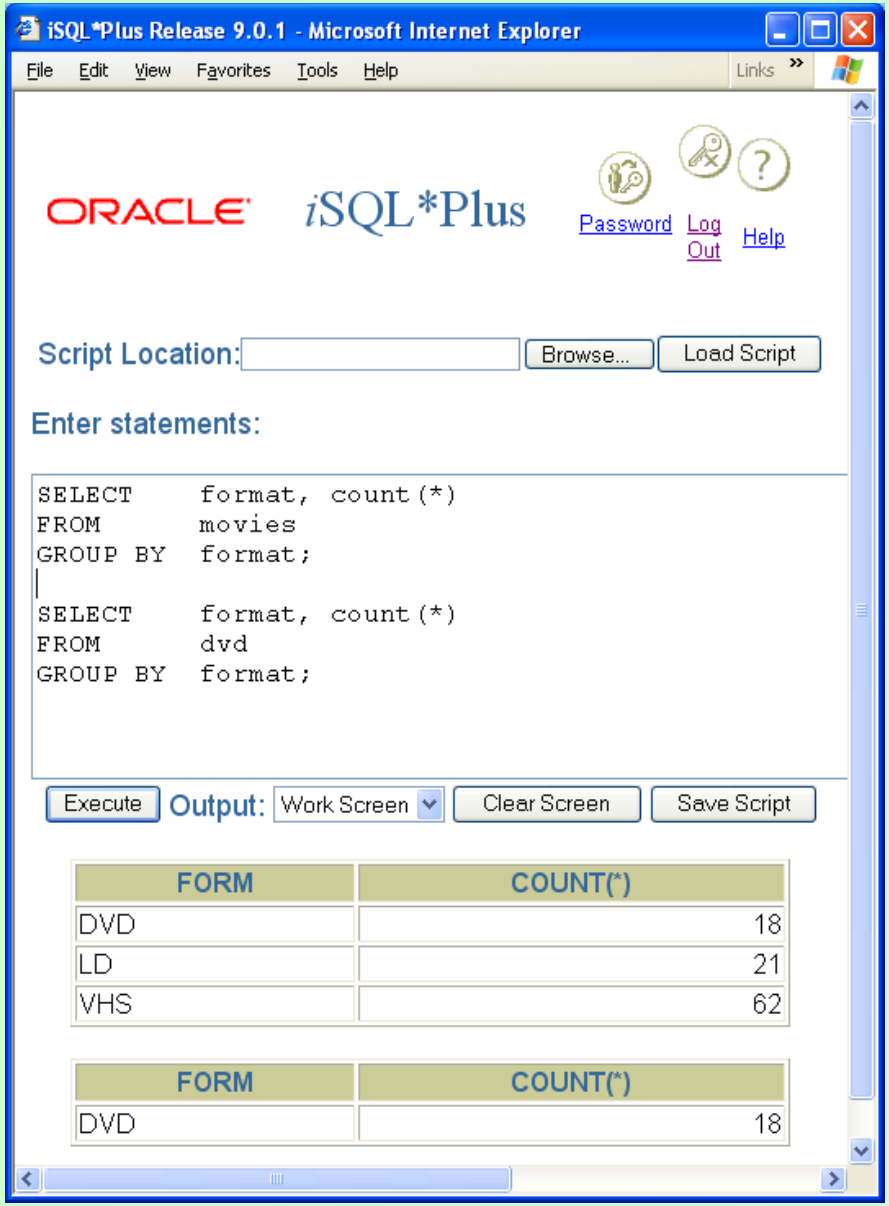
Step 2. Write the SELECT program

SELECT *
FROM movies
WHERE format = 'DVD'.

Put it all together

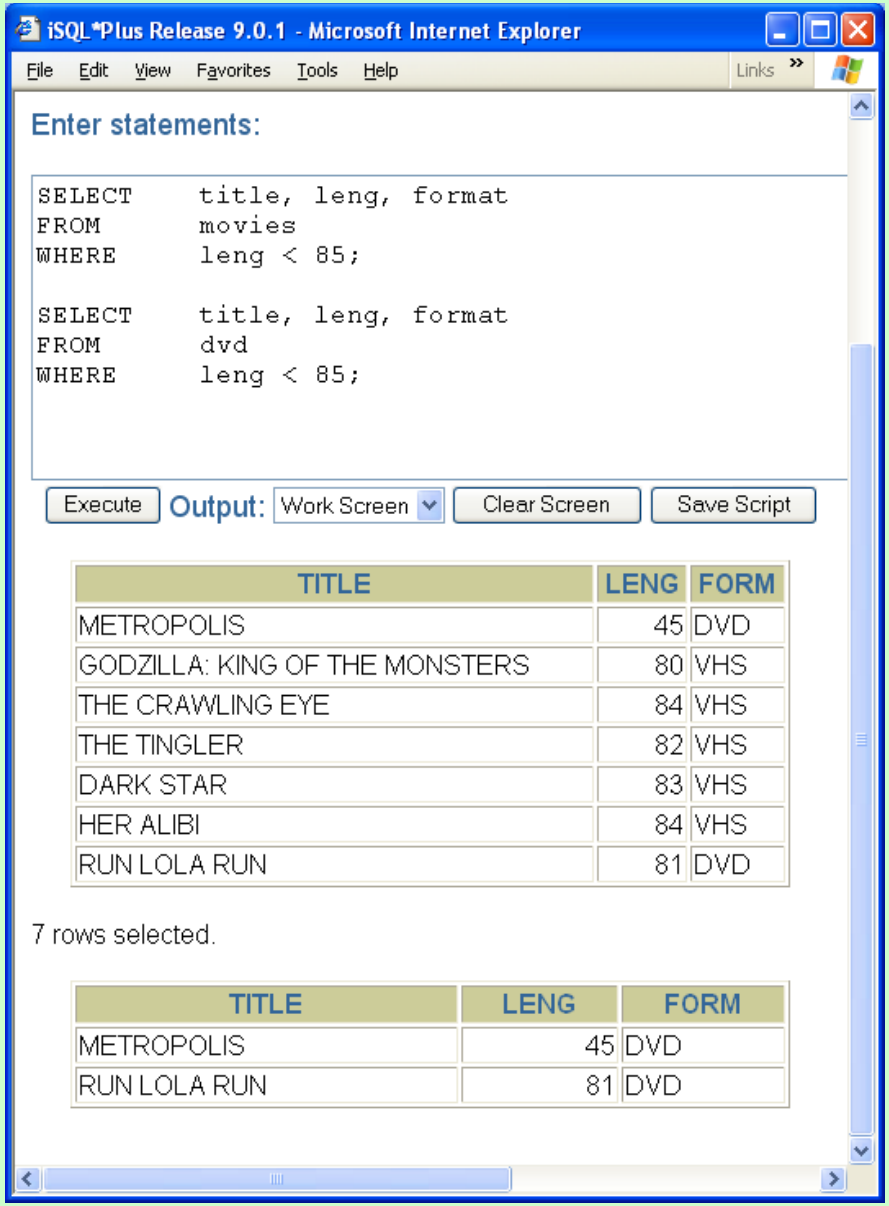
CREATE VIEW dvd AS
SELECT *
FROM movies
WHERE format = 'DVD';

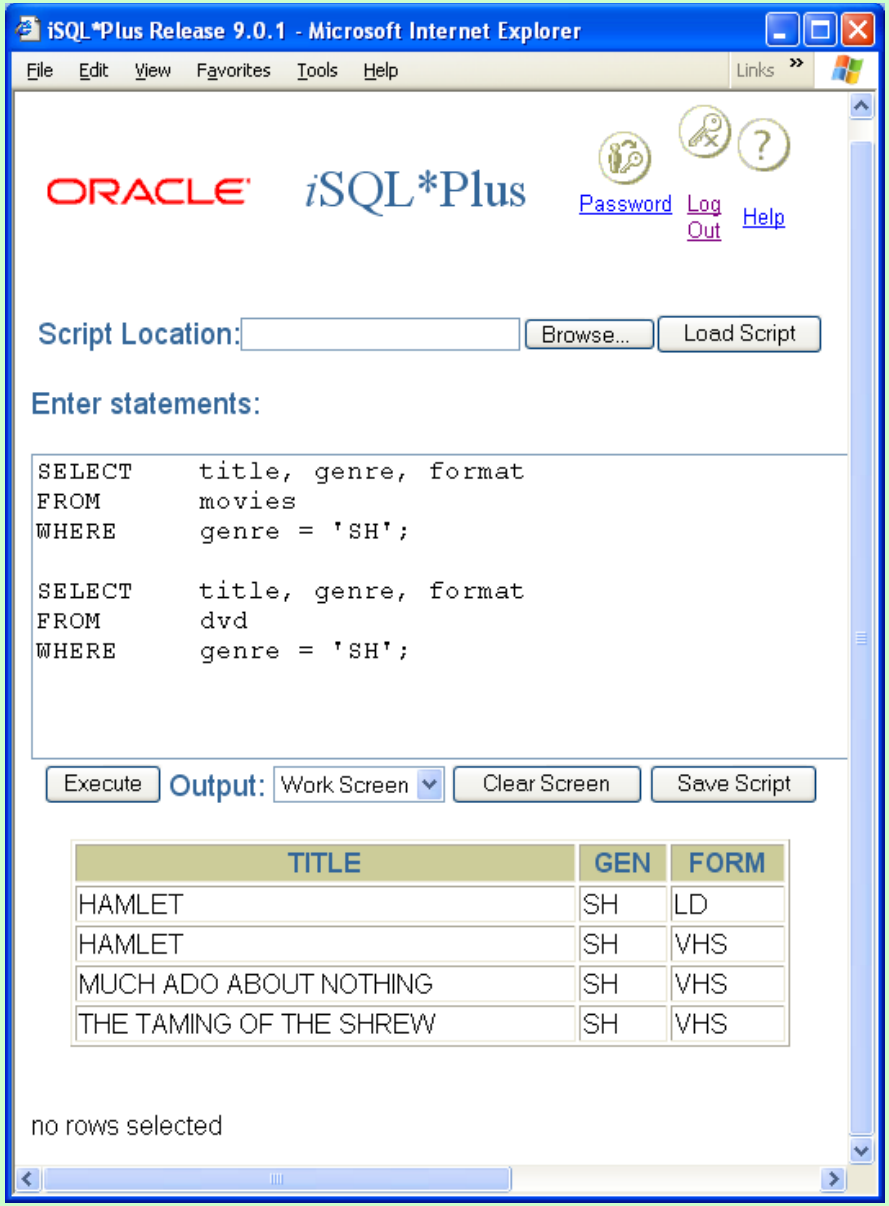
Page E-7 Using a view



This view should behave like any other table in the database.

Seems to have the same number of titles.





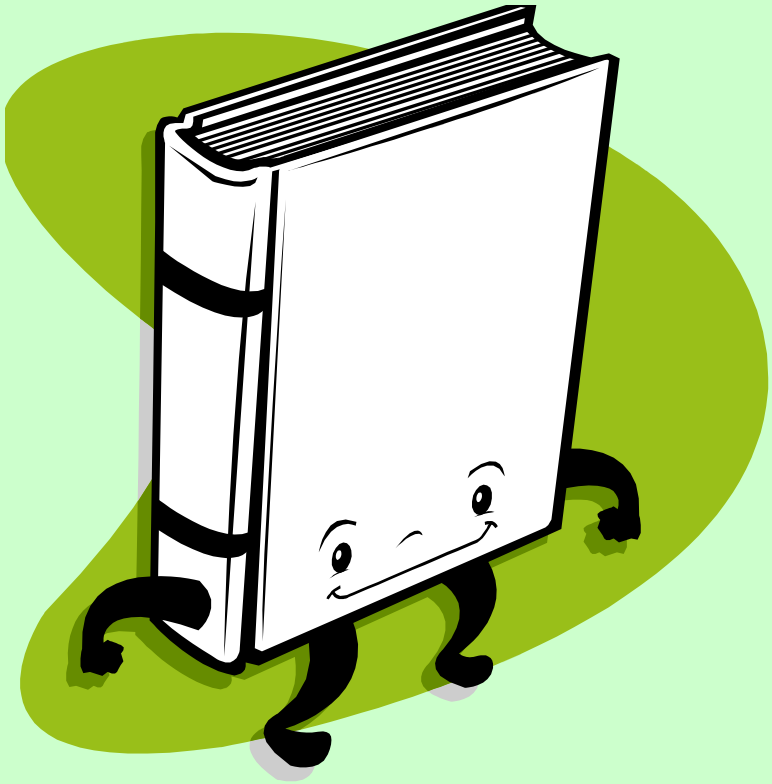
Page F-1: DROP

Structure are removed from the database with the DROP command.

The syntax for the DROP statement is something like this:

DROP *structure-type structure-name*

Page T-1: Terminology



CREATE
CREATE TABLE
CREATE VIEW

Datatype, domains
Numeric – NUMBER
Character – CHAR, VARCHAR2
Date – DATE

DROP
DROP TABLE
DROP VIEW

Please drop me an email if you noticed any errors in this module. I'd also appreciate reading your comments, criticisms, and or suggestions as to how this module could be improved.

Thanks,

bil

That’s All