We are going to make a simple table; insert a few rows and then run queries against the table. This is the table you are going to create for assignment 01. Since this is not part of a collection of related tables, I am going to create it in the a_testbed database.

The table will store data about animals- we will keep track of the name of each of our animals, the types of animal this is (Lion, Dog, Beetle, Centaur - all types of animals are welcome!); we want to know how much each animal costs; and we will keep the animal's date of birth and the date that we acquired the animal for our zoo. And we will have an id for each of our animals.

This series of attributes lets us experiment with the basic types of data we can store in a table- character data, numeric data and dates.

Don't worry that you do not understand all of this at the start; part of learning about manipulating computer systems is learning to follow a model. And I will supply all of the essential statements.

1. Creating the zoo_2016 table

With a database system, you need to create the place to store the data first; you need a database to store tables and you need to create tables to store data. With your Oracle account, you already have a database and schema so you just need to create the table.

We need a name for our table; since it stores animals for my zoo, I am going to name the table **zoo_2016**.

You need to provide names for the columns and specify the data types. In SQL this is done with a Create Table statement. This is the SQL to create the table. Pay attention to the punctuation marks- but you do not have to add spaces to align the words the way I have done here. The statement ends with a semicolon.

REMEMBER- you should have already run the provided script to create this table. If you run the script again, it will drop your table and recreate it.

```
create table zoo_2016
   (
     z_id       integer         not null
    , z_name     char( 25 )  null
    , z_type     char( 25 )  not null
    , z_cost     numeric( 7, 2 )  not null
    , z_dob      date            not null
    , z_acquired date            not null
   ) ;
```

1.1.1. Syntax for Create Table

Don't worry about this too much yet- but this is what that statement says.

"Create table" is a pair of SQL keywords that say that I want to define a table. I follow that with the name of the table.

Inside parentheses, we have a series of column definitions which supply a column name and a data type and a nullability clause. The data type **char** says that the column will store text; char(25) means that the text cannot be longer than 25 characters. The data type **integer** says that the column will store integer numbers; these are whole numbers that could be positive or negative. The data type **numeric**(7,2) says that the column will store decimal values that can contain up to 7 digits, 2 of these are after the decimal point. The data type **date** is a date.

If the column is marked as not null, then each row in the table has to have a value. The z_name column could be left empty

If you have previous experience with SQL tables you might notice that the table does not have a primary key. We will talk more about this in the next unit, but in the assignment people supply sample inserts and if the table had a primary key, then we would need a way to make sure that each inserted row had a different value for the primary key column.

Experiment: When you do the inserts, try inserting data that breaks the rules in the Create table statement. What happens?

1.1.2. What are the names of my tables?

If you want to see the names of the tables you have, you can use

```
SELECT *
FROM INFORMATION_SCHEMA.TABLES table_name;
```

1.1.3. What is the structure of my table?

Use this SQL command to see the columns in your table.

```
Select column_name
From INFORMATION_SCHEMA.COLUMNS
Where TABLE_NAME = 'zoo_2016'

 Column_name
 ------------
 Z_ID
 Z_NAME
 Z_TYPE
 Z_COST
 Z_DOB
 Z_ACQUIRED
```

2. Insert statement

The second script inserted rows into the table. This is done with an insert statement. The column names in the first clause match up with the data values. You need the parentheses and the commas between items and you need the single quotes around the character values. Note that we do not put quotes around the numeric values.

The date values are being entered using a format; for now, just follow that pattern. (You should have already run the insert script; if you run it again it will empty the rows from the table and then insert them again.)

This is a copy of the first insert in the script. Do not run it again here.

```
Insert Into zoo_2016  (z_id, z_name, z_type, z_cost, z_dob, z_acquired)
Values
     (23, 'Sam', 'Giraffe', 5000.00, '2014-05-15','2014-05-15');;
```

The response with each of these inserts is "1 row created". We do not get a display of the data because we did not ask for one.

3. Select statement

Now that we have a table and have entered some rows, we could ask what is in our table.

That is done with a Select statement. Enter this statement in your client.

Demo 01: Select query to see the data in a table.

```
select z_id, z_name, z_type, z_cost, z_dob, z_acquired
from zoo_2016;
```

We now see the data

```
      Z_ID Z_NAME     Z_TYPE                        Z_COST Z_DOB     Z_ACQUIRE
---------- ---------- ------------------------- ---------- --------- ---------
        23 Sam        Giraffe                         5000 15-MAY-14 15-MAY-14
        25 Abigail    Armadillo                        490 15-JAN-13 15-APR-13
        56 Leon       Lion                            5000 25-FEB-10 25-MAR-10
        57 Lenora     Lion                            5000 25-MAR-14 31-MAR-14
        85 Sally Robi Giraffe                      5000.25 15-MAY-14 15-MAR-14
        43 Huey       Zebra                        2500.25 02-JUN-13 02-JUN-14
        44 Dewey      Zebra                        2500.25 02-JUN-14 02-JUN-14
        45 Louie      Zebra                        2500.25 02-JAN-13 02-JAN-13
        47            Horse                            490 10-JAN-15 15-JAN-15
        52 Dewey      Giraffe                         3750 06-JUN-13 12-JUL-13

10 rows selected.
```

4. Filter

We might want to see only some of the animals. For the output displays I sometimes narrow the columns to fit the page.

Demo 02: --we want to see the armadillos; For Oracle the literals have to match in Case. If I filters for 'armadillo' I do not get any rows returned.

```
select z_id, z_name, z_type, z_cost
from zoo_2016
where z_type = 'Armadillo'
;
```

```
      Z_ID Z_NAME                    Z_TYPE                        Z_COST
---------- ------------------------- ------------------------- ----------
        25 Abigail                   Armadillo                        490
```

Demo 03: We want to see animals that cost 5000

```
select z_id, z_name, z_type, z_cost
from zoo_2016
where z_cost = 5000
;
```

```
      Z_ID Z_NAME                    Z_TYPE                         Z_COST
---------- ------------------------- ------------------------- ----------
        23 Sam                       Giraffe                         5000
        56 Leon                      Lion                            5000
        57 Lenora                    Lion                            5000
```

Demo 04: We want to see the unicorns- but I do not have any

```
select z_id, z_name, z_type, z_cost
from zoo_2016
where z_type = 'Unicorn';
```

```
no rows selected
```

This has been a quick tour through the main types of SQL statements: a statement to create a structure to hold data; a statement to modify (in this case add) the data in the table; and statements to display the data in the table.

5. The Insert statement

The Insert statement has a few variations. The table zoo_2016 has six columns. The column names are listed in the first parenthesized list- the column list; and the values are listed in the same order in the values list. It is possible to change the order of the columns in the column list - although that is not generally useful.

Demo 05: More inserts- you should run these

```
Insert Into zoo_2016 (
  z_id
, z_name
, z_type
, z_cost
, z_dob
, z_acquired
) VALUES
(
  257
,'Arnold'
,'Giraffe'
, 5000.00
, '2014-05-15'
, '2014-05-15'
);
```

If you did not want to supply a value for the animal name (it is not a required column), you could skip that column in the column list and skip an entry in the values list.

```
Insert Into zoo_2016 (
  z_id
, z_type
, z_cost
, z_dob
, z_acquired
) VALUES
(
  258
,'Giraffe'
, 5000.00
, '2013-05-15'
, '2013-05-31'
);
```

If you supply a value for every column in the table and you supply the values in the same order as they are listed in the Create table statement, then you are allowed to skip the column list. Here I am using the value null for the animal name.

```
Insert Into zoo_2016 VALUES
(
  259
, null
,'Giraffe'
, 5000.00
, '2002-05-15'
, '2002-05-15'
);
```

Now we have a herd of giraffe.

```
      Z_ID Z_NAME     Z_TYPE         Z_COST Z_DOB     Z_ACQUIRE
---------- ---------- ---------- ---------- --------- ---------
        23 Sam        Giraffe          5000 15-MAY-14 15-MAY-14
        85 Sally Robi Giraffe       5000.25 15-MAY-14 15-MAR-14
        52 Dewey      Giraffe          3750 06-JUN-13 12-JUL-13
       257 Arnold     Giraffe          5000 15-MAY-14 15-MAY-14
       258            Giraffe          5000 15-MAY-13 31-MAY-13
       259            Giraffe          5000 15-MAY-02 15-MAY-02


7 rows selected.
```