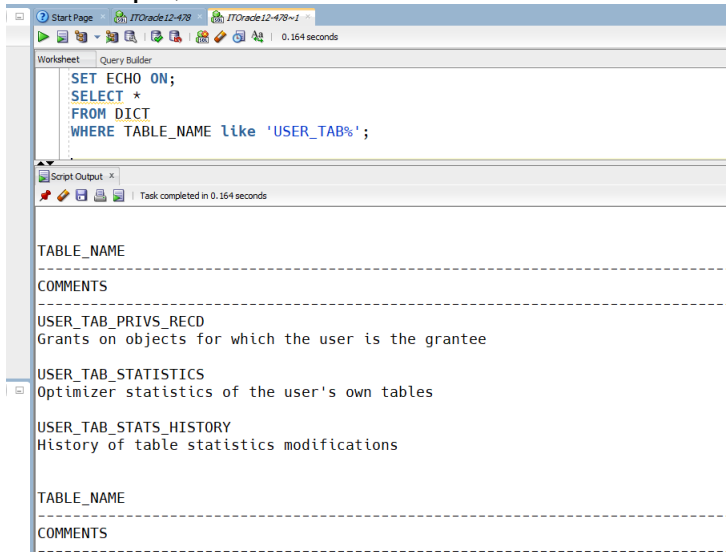


Lab #9 Assignment Tips

As some of you have noticed, a few of the queries are UGLY!

For example,



This ugly output is caused by some of the very large column sizes of the systems tables we are querying.

Let's look at the **DICT**ionary table

```
SQL> DESC dict;
Name          Null? Type
-----
TABLE_NAME    VARCHAR2(128)
COMMENTS      VARCHAR2(4000)
```

We can use the **SHOW LINESIZE** to see that our line is much smaller than our data

```
SQL> SHOW LINESIZE;
linesize 1566
```

My LINESIZE is set to 1566. Your LINESIZE may be even smaller. You can test it to find out.

So, what should we do to “de-uglify” our output.

Option 1: Nothing. Really, this is perfectly acceptable. The assignment is about exploring the `USER_*` and `ALL_*` tables, not about “prettifying” your output. BUT, I know many of you will reject this option.

Option 2: Format the Columns. This is an SQL*PLUS trick. But, it also works with **SQL Developer**. In the example below, I have formatted the **TABLE_NAME** and the **COMMENTS** columns to be 20 characters long (A20). NOTE: I needed a format command for each column I wanted to format, and that I **did not use semicolons** at the end of the lines.

The screenshot shows the SQL Developer interface. In the Query Builder, the following SQL*PLUS commands are entered:

```

COLUMN TABLE_NAME FORMAT A20
COLUMN COMMENTS FORMAT A20
SELECT *
FROM DICT
WHERE TABLE_NAME like 'USER_TAB%';

```

Two red arrows point to the `COLUMN` commands. Below the query, the Script Output and Query Result panes show the execution results. The Query Result pane displays a table with two columns: `TABLE_NAME` and `COMMENTS`. The output is as follows:

TABLE_NAME	COMMENTS
USER_TABLES	Description of the user's own relational tables
USER_TABLESPACES	Description of accessible tablespaces
USER_TAB_COLS	Columns of user's tables, views and clusters

A red arrow points to the `USER_TAB_COLS`

But, the word wrapping in the **COMMENTS** column is a bit ugly.

I can fix this with the **WORD_WRAPPED** command. See below.

The screenshot shows the SQL Developer interface. In the Query Builder, the following SQL*PLUS commands are entered:

```

COLUMN TABLE_NAME FORMAT A20
COLUMN COMMENTS FORMAT A20 WORD_WRAPPED
SELECT *
FROM DICT
WHERE TABLE_NAME like 'USER_TAB%';

```

A red arrow points to the `WORD_WRAPPED` command. Below the query, the Script Output and Query Result panes show the execution results. The Query Result pane displays a table with two columns: `TABLE_NAME` and `COMMENTS`. The output is as follows:

TABLE_NAME	COMMENTS
USER_TAB_PRIVS_RECD	Grants on objects for which the user is the grantee
USER_TAB_STATISTICS	Optimizer statistics of the user's own tables
USER_TAB_STATS_HISTORY	History of table statistics

NOTE: The **CLEAR COLUMNS** command clears the above column formatting.

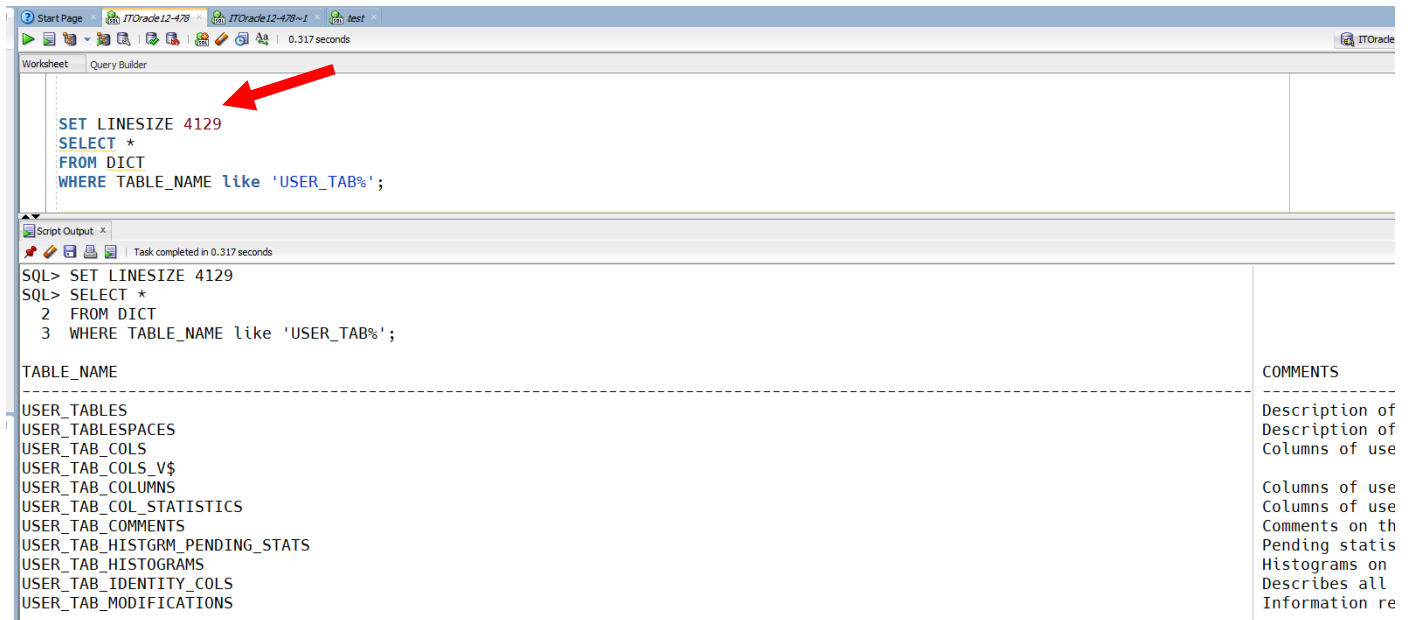
Option 3: Change the **LINESIZE**.

```
SQL> DESC dict;
```

Name	Null?	Type
TABLE_NAME		VARCHAR2(128)
COMMENTS		VARCHAR2(4000)

128 + 1 + 4000 = 4129 (There is a space between each column)

So, setting the **LINESIZE** solves the ugliness problem (sort of). Again, this line does not need a semicolon. See below.



The screenshot shows the SQL Developer interface. In the 'Query Builder' tab, the following SQL query is entered:

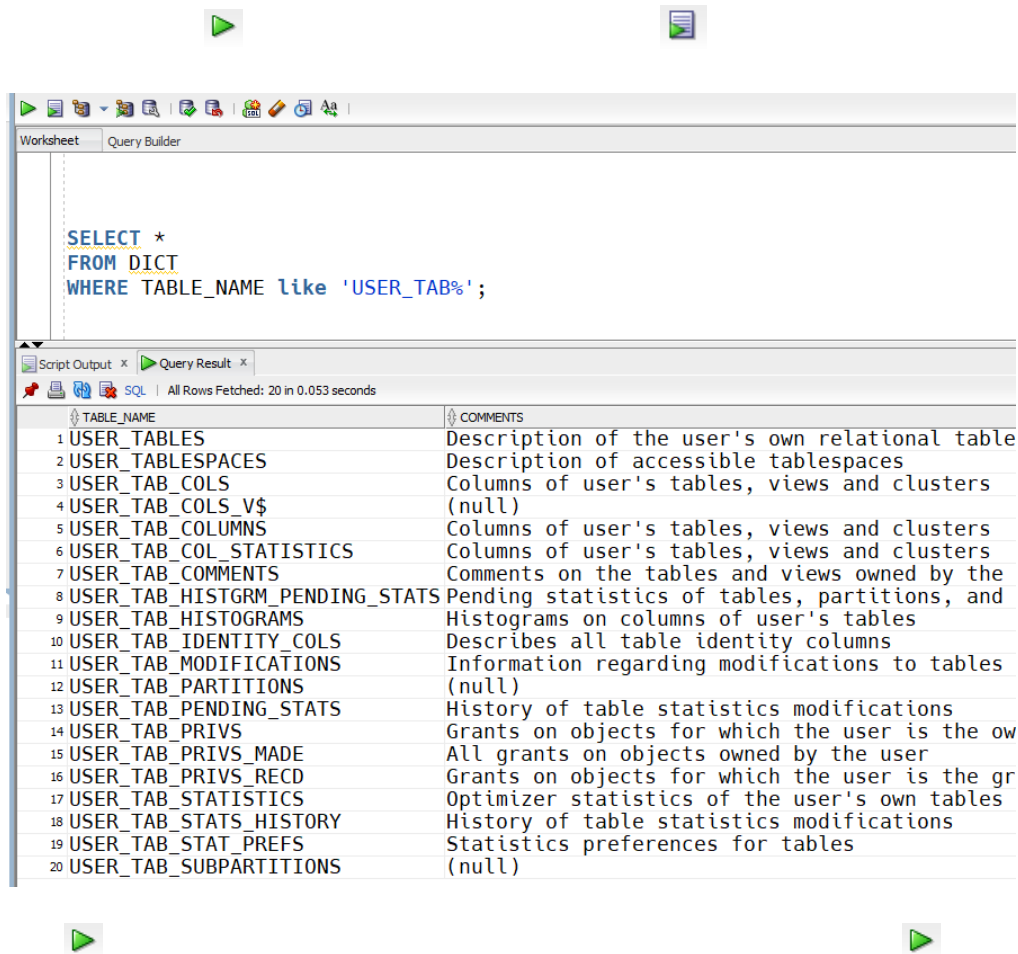
```
SET LINESIZE 4129
SELECT *
FROM DICT
WHERE TABLE_NAME like 'USER_TAB%';
```

A red arrow points to the `SET LINESIZE 4129` command. Below the query editor, the 'Script Output' window shows the execution results:

```
SQL> SET LINESIZE 4129
SQL> SELECT *
2  FROM DICT
3  WHERE TABLE_NAME like 'USER_TAB%';
```

The output is displayed as a table with two columns: **TABLE_NAME** and **COMMENTS**.

TABLE_NAME	COMMENTS
USER_TABLES	
USER_TABLESPACES	
USER_TAB_COLS	
USER_TAB_COLS_V\$	
USER_TAB_COLUMNS	
USER_TAB_COL_STATISTICS	
USER_TAB_COMMENTS	
USER_TAB_HISTGRM_PENDING_STATS	
USER_TAB_HISTOGRAMS	
USER_TAB_IDENTITY_COLS	
USER_TAB_MODIFICATIONS	



The screenshot shows the Oracle SQL Developer interface. At the top, there's a toolbar with various icons. Below it, the 'Worksheet' tab is active, displaying a SQL query:

```
SELECT *
FROM DICT
WHERE TABLE_NAME like 'USER_TAB%';
```

Below the query editor, the 'Query Result' tab is active, showing the results of the query. The results are displayed in a table with two columns: 'TABLE_NAME' and 'COMMENTS'. The table contains 20 rows of data, listing various system tables and their descriptions.

TABLE_NAME	COMMENTS
1 USER_TABLES	Description of the user's own relational table
2 USER_TABLESPACES	Description of accessible tablespaces
3 USER_TAB_COLS	Columns of user's tables, views and clusters
4 USER_TAB_COLS_V\$	(null)
5 USER_TAB_COLUMNS	Columns of user's tables, views and clusters
6 USER_TAB_COL_STATISTICS	Columns of user's tables, views and clusters
7 USER_TAB_COMMENTS	Comments on the tables and views owned by the
8 USER_TAB_HISTGRM_PENDING_STATS	Pending statistics of tables, partitions, and
9 USER_TAB_HISTOGRAMS	Histograms on columns of user's tables
10 USER_TAB_IDENTITY_COLS	Describes all table identity columns
11 USER_TAB_MODIFICATIONS	Information regarding modifications to tables
12 USER_TAB_PARTITIONS	(null)
13 USER_TAB_PENDING_STATS	History of table statistics modifications
14 USER_TAB_PRIVS	Grants on objects for which the user is the ow
15 USER_TAB_PRIVS_MADE	All grants on objects owned by the user
16 USER_TAB_PRIVS_RECD	Grants on objects for which the user is the gr
17 USER_TAB_STATISTICS	Optimizer statistics of the user's own tables
18 USER_TAB_STATS_HISTORY	History of table statistics modifications
19 USER_TAB_STAT_PREFS	Statistics preferences for tables
20 USER_TAB_SUBPARTITIONS	(null)

Final note: If you do not get any results for the last question, it is probably because you didn't do the first question correctly.

Here is the text from question 1:

1. a. Use SQL to create the tables **ch10_suppliers** and **ch10_subjects**. Once created, the two tables should have the same column names, null? and type listed below. In addition, **SUBJECT_ID**, **SUBJECT_NAME** and **SUPPLIER_ID** should all be created with the **UNIQUE** constraint.

A few people have missed this highlighted part. If you add the **UNIQUE** constraints for the three columns. **You will get results for the final question.**

In Oracle 12c, users can create indexes with the CREATE INDEX command), but Oracle automatically creates indexes for **UNIQUE** and **PRIMARY KEY** constraints. I will be talking more about constraints and indexes in future lectures.