1. [1] List the rowid of *‘A01*’in your *Authors* table. List the *object\_id* of your *Authors* table. Write a SQL command to calculate the integer value of the rowid to verify it belongs to the object id.

SELECT ROWID,

SUBSTR(ROWID,1,6) AS OBJECT\_ID,

(0 \* POWER(64,5)+

0 \* POWER(64,4)+

0 \* POWER(64,3)+

22 \* POWER(64,2)+

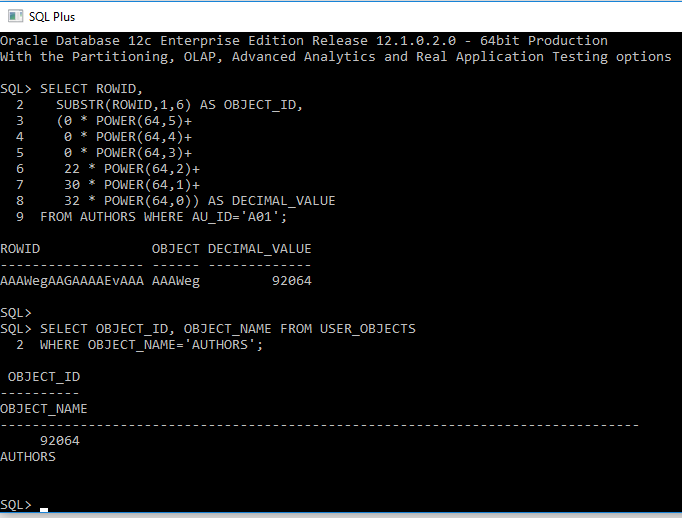
30 \* POWER(64,1)+

32 \* POWER(64,0)) AS DECIMAL\_VALUE

FROM AUTHORS WHERE AU\_ID='A01';

SELECT OBJECT\_ID, OBJECT\_NAME FROM USER\_OBJECTS

WHERE OBJECT\_NAME='AUTHORS';



1. [2] List the *title\_id, title, units sold, current price* and proposed new 20% higher price of all books. Label the proposed new price column "*Target Price*". Sort the results in descending incremental dollar amount.

SELECT TITLE\_ID, TITLE, SALES AS UNITS\_SOLD, PRICE, PRICE\*1.2 AS TARGET\_PRICE FROM TITLES

ORDER BY (PRICE\*0.2) DESC;



1. [2] List *au\_id, fname, lname* and *number\_of\_books\_sold* of the author that has written the most books.

SELECT AU.AU\_ID, AU.FNAME, AU.LNAME, AUTIT.TITLE\_NUMBERS AS UNITS\_SOLD FROM AUTHORS AU

INNER JOIN

(

SELECT DISTINCT AU\_ID,

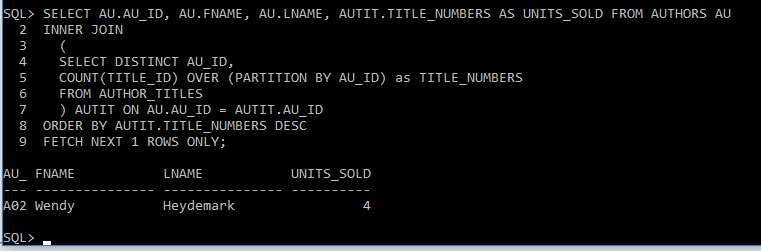
COUNT(TITLE\_ID) OVER (PARTITION BY AU\_ID) as TITLE\_NUMBERS

FROM AUTHOR\_TITLES

) AUTIT ON AU.AU\_ID = AUTIT.AU\_ID

ORDER BY AUTIT.TITLE\_NUMBERS DESC

FETCH NEXT 1 ROWS ONLY;



1. [3] List the title\_id and title of every book that contains the word '*the*', in any case. For example, “*The house at …”, and “… the road ahead”,* but not *“…another…”.* Sort the results in increasing order according to the location of the *'the'* word. *Hint: Oracle has a function that returns location of a substring within a text string*.

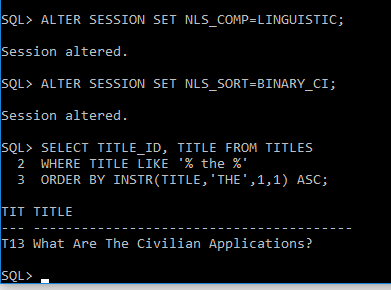
ALTER SESSION SET NLS\_COMP=LINGUISTIC;

ALTER SESSION SET NLS\_SORT=BINARY\_CI;

SELECT TITLE\_ID, TITLE FROM TITLES

WHERE TITLE LIKE '% the %'

ORDER BY INSTR(TITLE,'THE',1,1) ASC;



1. [3] List revenue by (*book type* and *publisher*), by (*publisher* and *year*)*,* and overall revenue. Sort by *publisher*.

SELECT T1.GENRE AS BOOK\_TYPE, T1.PUB\_ID AS PUBLISHER, SUM(T1.REVENUE) AS OVERALL\_REVENUE FROM

(

SELECT T.TITLE\_ID, T.GENRE, NVL(ROUND(T.SALES\*T.PRICE),0) AS REVENUE, NVL(T.PUB\_ID, '') AS PUB\_ID, T.PUB\_GROUP,

ROW\_NUMBER() OVER (PARTITION BY T.GENRE ORDER BY NVL(T.PUB\_ID, '') ASC) as GENRE\_GROUP

FROM

(

SELECT

TITLE\_ID, GENRE, PRICE, SALES, NVL(PUB\_ID,'') AS PUB\_ID,

ROW\_NUMBER() OVER (PARTITION BY PUB\_ID ORDER BY NVL(PUB\_ID,'') ASC) as PUB\_GROUP

FROM TITLES

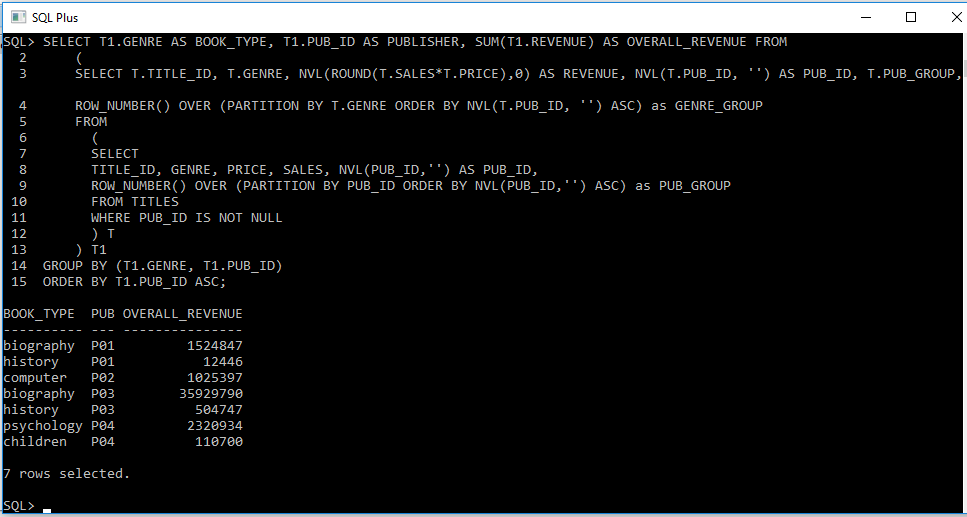
WHERE PUB\_ID IS NOT NULL

) T

) T1

GROUP BY (T1.GENRE, T1.PUB\_ID)

ORDER BY T1.PUB\_ID ASC;



SELECT T1.PUB\_ID AS PUBLISHER, T1.PUB\_DATE AS "YEAR", SUM(T1.REVENUE) AS OVERALL\_REVENUE FROM

(

SELECT T.TITLE\_ID, '20'||SUBSTR(T.PUBDATE,8,2) AS PUB\_DATE, NVL(ROUND(T.SALES\*T.PRICE),0) AS REVENUE, NVL(T.PUB\_ID, '') AS PUB\_ID, T.PUB\_GROUP,

ROW\_NUMBER() OVER (PARTITION BY SUBSTR(T.PUBDATE,8,2) ORDER BY NVL(T.PUB\_ID, '') ASC) as GENRE\_GROUP

FROM

(

SELECT

TITLE\_ID, PUBDATE, PRICE, SALES, NVL(PUB\_ID,'') AS PUB\_ID,

ROW\_NUMBER() OVER (PARTITION BY PUB\_ID ORDER BY NVL(PUB\_ID,'') ASC) as PUB\_GROUP

FROM TITLES

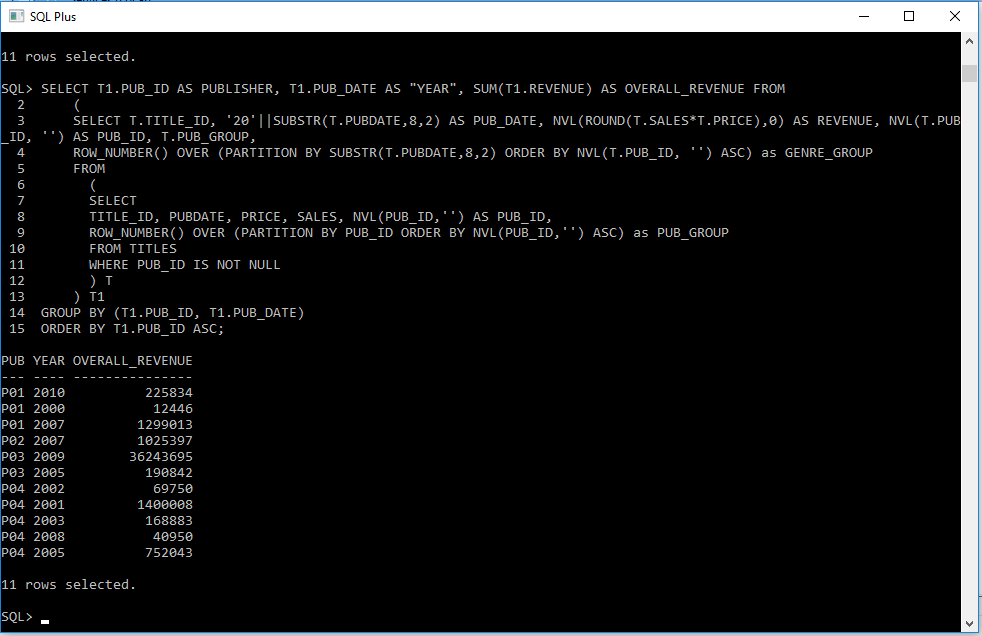
WHERE PUB\_ID IS NOT NULL

) T

) T1

GROUP BY (T1.PUB\_ID, T1.PUB\_DATE)

ORDER BY T1.PUB\_ID ASC;



1. [3] List the *title\_id, revenue,* and *pub\_id* of the book or books that have the greatest revenue for each *pub\_id*.

SELECT T.TITLE\_ID, T.REVENUE, T.PUB\_ID FROM

(SELECT

TITLE\_ID, GENRE, PRICE, SALES, NVL(PUB\_ID,'') AS PUB\_ID, NVL(ROUND(SALES\*PRICE),0) AS REVENUE,

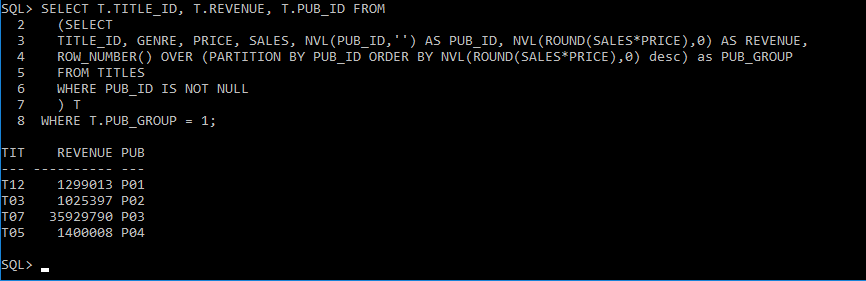
ROW\_NUMBER() OVER (PARTITION BY PUB\_ID ORDER BY NVL(ROUND(SALES\*PRICE),0) desc) as PUB\_GROUP

FROM TITLES

WHERE PUB\_ID IS NOT NULL

) T

WHERE T.PUB\_GROUP = 1;



1. [3] Create the following results using a materialized view:

GRANT CREATE MATERIALIZED VIEW TO carlos;

GRANT QUERY REWRITE TO carlos;

GRANT CREATE ANY TABLE TO carlos;

CREATE MATERIALIZED VIEW MV\_TITLES AS

SELECT INITCAP(GENRE) GENRE,

AVG(SALES) AS AVERAGE\_SALES,

STDDEV(SALES) AS STDDEV\_SALES

FROM titles

WHERE GENRE IS NOT NULL

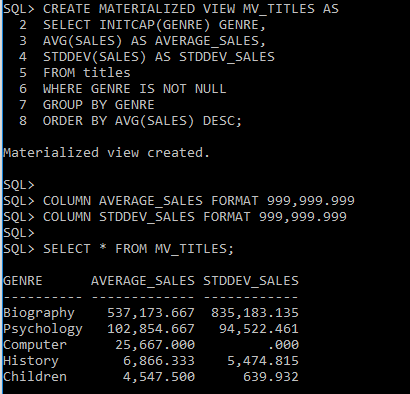
GROUP BY GENRE

ORDER BY AVG(SALES) DESC;

COLUMN AVERAGE\_SALES FORMAT 999,999.999

COLUMN STDDEV\_SALES FORMAT 999,999.999

SELECT \* FROM MV\_TITLES;



[3] Use the materialized view created above to generate the following results:

SELECT T.TITLE\_ID,

INITCAP(T.GENRE) as GENRE,

T.SALES,

MVT.AVERAGE\_SALES,

MVT.STDDEV\_SALES,

(MVT.AVERAGE\_SALES+MVT.STDDEV\_SALES) as "Sales Plus StdDev"

FROM TITLES T

INNER JOIN MV\_TITLES MVT ON T.GENRE = MVT.GENRE

WHERE T.SALES > MVT.AVERAGE\_SALES+MVT.STDDEV\_SALES

ORDER BY SALES DESC;

COLUMN SALES HEADING SALES JUSTIFY CENTER FORMAT 999,999,999

COLUMN TITLE\_ID HEADING 'Title|Id' JUSTIFY CENTER FORMAT A8

COLUMN Genre HEADING GENRE JUSTIFY CENTER

COLUMN STDDEV\_SALES HEADING 'Stddev|Sales' JUSTIFY CENTER FORMAT 999,999.99

COLUMN AVERAGE\_SALES HEADING 'Avg|Sales' JUSTIFY CENTER FORMAT 999,999.99

COLUMN 'Sales Plus StdDev' HEADING 'Sales plus|Stddev' JUSTIFY CENTER FORMAT 9,999,999.99

