--AGYEMANG ERIC

--IT 478 ADVANCED DATABASE PROCESSING

-- FINAL EXAMS Part 2

--14.Write a procedure to grant user IT47822 SELECT, INSERT, UPDATE, and DELETE privileges on the SETS, COLORS and THEMES table.

SET ECHO ON;

--Procedure to grant user IT47822 privileges on the SETS table

CREATE OR REPLACE PROCEDURE grant\_user\_priv\_sets

(

sets VARCHAR2

)

AS

BEGIN

EXECUTE IMMEDIATE 'GRANT SELECT, INSERT, UPDATE, DELETE ON sets TO IT47822';

EXCEPTION

WHEN OTHERS THEN

NULL;

END;

call grant\_user\_priv\_sets('SETS')

--Procedure to grant user IT47822 privileges on the COLORS table

CREATE OR REPLACE PROCEDURE grant\_user\_priv\_colors

(

colors VARCHAR2

)

AS

BEGIN

EXECUTE IMMEDIATE 'GRANT SELECT, INSERT, UPDATE, DELETE ON colors TO IT47822';

EXCEPTION

WHEN OTHERS THEN

NULL;

END;

call grant\_user\_priv\_colors('COLORS')

--Procedure to grant user IT47822 privileges on the THEMES table.

CREATE OR REPLACE PROCEDURE grant\_user\_priv\_themes

(

themes VARCHAR2

)

AS

BEGIN

EXECUTE IMMEDIATE 'GRANT SELECT, INSERT, UPDATE, DELETE ON themes TO IT47822';

EXCEPTION

WHEN OTHERS THEN

NULL;

END;

call grant\_user\_priv\_themes('THEMES')

--15.Query the USER\_TAB\_PRIVS table to show only user IT47822’s privileges

COLUMN TABLE\_NAME FORMAT A20

COLUMN GRANTEE FORMAT A20

SELECT table\_name, grantee, privilege

FROM USER\_TAB\_PRIVS

WHERE Grantee = 'IT47822';

--16.Write a procedure to revoke all user IT47822 privileges on the SETS table.

CREATE OR REPLACE PROCEDURE all\_revok\_user\_priv

(

sets VARCHAR2

)

AS

BEGIN

EXECUTE IMMEDIATE 'REVOKE ALL ON sets FROM IT47822';

EXCEPTION

WHEN OTHERS THEN

NULL;

END;

call all\_revok\_user\_priv('SETS')

--17.Query the USER\_TAB\_PRIVS table to show all privileges

COLUMN TABLE\_NAME FORMAT A20

COLUMN GRANTEE FORMAT A20

SELECT table\_name, grantee, privilege

FROM USER\_TAB\_PRIVS;

--18.Query the proper systems table to show all constraints on the part\_categories table.

COLUMN OWNER FORMAT A15

COLUMN CONSTRAINT\_NAME FORMAT A20

COLUMN CONSTRAINT\_TYPE FORMAT A15

COLUMN TABLE\_NAME FORMAT A15

COLUMN COLUMN\_NAME FORMAT A20

COLUMN SEARCH\_CONDITION FORMAT A15

COLUMN SEARCH\_CONDITION\_VC FORMAT A5

COLUMN LAST\_CHANGE FORMAt A20

COLUMN INDEX\_OWNER FORMAt A20

COLUMN INDEX\_NAME FORMAT A10

-- Using \* to display the output

SELECT \*

FROM ALL\_CONSTRAINTS

WHERE table\_name = 'PART\_CATEGORIES';

-- using specific column names to display the output

COLUMN OWNER FORMAT A15

COLUMN CONSTRAINT\_NAME FORMAT A20

COLUMN CONSTRAINT\_TYPE FORMAT A15

COLUMN TABLE\_NAME FORMAT A15

COLUMN COLUMN\_NAME FORMAT A20

COLUMN SEARCH\_CONDITION FORMAT A15

COLUMN SEARCH\_CONDITION\_VC FORMAT A5

COLUMN LAST\_CHANGE FORMAt A20

COLUMN INDEX\_OWNER FORMAt A20

COLUMN INDEX\_NAME FORMAT A10

SELECT owner, constraint\_name, constraint\_type,table\_name, index\_name , last\_change

FROM ALL\_CONSTRAINTS

WHERE TABLE\_NAME='PART\_CATEGORIES';

--19.Create the query needed to answer: How may blue parts are there in the Lego data?

SELECT COUNT(inventory\_id) AS Numer\_of\_Blue\_Parts

FROM inventory\_parts ip JOIN colors c ON ip.color\_id =c.id

WHERE c.name = 'Blue';

--20.Create the query needed to answer: What are the newest sets in the Lego data?

SELECT year, name AS Newest\_Set\_Name, set\_num AS Newest\_set\_num, num\_parts

FROM sets

where year = (select MAX(year) from sets);

--21.Create the query needed to answer: What are the newest sets in the Lego data WITH a Star Wars theme?

SELECT year, s.name AS Set\_Name, set\_num, t.name AS theme\_name

FROM sets s JOIN themes t ON s.theme\_id = t.id

where year = (SELECT MAX(year) from sets) AND t.name = 'Star Wars';

--22.Find average number of pieces in each Lego set (by year) for the years 2010-2015. Order the results from highest to lowest

SELECT year, ROUND(AVG(num\_parts), 2) AS Average\_number\_of\_pieces

FROM sets

GROUP BY year

HAVING year BETWEEN 2010 AND 2015

ORDER BY year DESC;

--23.Create the query needed to answer: Which year in the 1980’s had the most sets?

SELECT year AS year\_with\_most\_sets

FROM sets

GROUP BY year

HAVING COUNT(set\_num) = (SELECT MAX(COUNT(\*))

FROM sets

WHERE year BETWEEN 1980 AND 1989

GROUP BY year);

--24.Create the query needed to answer: Which theme was the most popular in the 1980’s?

SELECT th.name AS most\_popular\_theme

FROM themes th JOIN sets st ON th.id = st.theme\_id

WHERE year BETWEEN 1980 AND 1989

GROUP BY th.name

HAVING COUNT(th.name) =(SELECT MAX(COUNT(t.name))

FROM themes t JOIN sets s ON t.id = s.theme\_id

WHERE year BETWEEN 1980 AND 1989

GROUP BY t.name);

--25. Write a function that will give the number of days between a provided date and February 1, 2022. For example, if you provided the day 11/15/2021, your function should return 78. Use the dual table to test your function.

CREATE OR REPLACE FUNCTION get\_number\_days

(

provided\_date\_param DATE

)

RETURN INTEGER

AS

number\_of\_days\_between INTEGER;

BEGIN

SELECT abs (TO\_DATE('01-feb-2022') - provided\_date\_param) AS "Number of days"

INTO

number\_of\_days\_between

FROM DUAL;

RETURN

number\_of\_days\_between;

END;

/

Function GET\_NUMBER\_DAYS compiled

SQL> SELECT get\_number\_days('15-nov-2021') AS "Number of days" FROM dual;

**--Answer the following completely (in your own words)**

**--26. Explain the difference between procedure and trigger.**

--The difference is that a procedure is a database object that contains a block of PL/SQL code. A --stored procedure can be used to modify the data that is stored within database. Example, it can

--be used to execute an INSERT, UPDATE, or DELETE statement. IF a procedure accepts more --than one parameter, you must use a comma to separate each parameter. A trigger on the other

--hand is a named block of PL/SQL code that is executed or fired automatically when a particular --type of SQL statement is executed. A trigger is fired mostly when an INSERT, UPDATE, or

--DELETE statement is executed on a table. IT is also possible to create a trigger that is fired when --a DDL statement such as CREATE, ALTER, or DROP statement is executed.

**--27. What are the 3 types of parameters in a PL/SQL procedure?**

--They are 1) IN type parameter

--2) OUT type parameter

--3) IN OUT parameter

--An IN formal parameter is initialized to the actual parameter with which it was called unless it was explicitly initialized with a default value.

--A OUT formal parameter is initialized to the actual parameter with which it was called. The called program can reference and assign new values to the formal parameter.

--An IN OUT formal parameter is initialized to the actual parameter with which it was called. It is modified by the called program, and the last value of the formal parameter is passed to the calling program’s actual parameter if the called program terminates without an exception.

--28. What is the dual table?

This is a special one-row, one-column table present by default in oracle and other databases installations. In oracle, the table has a single a single VARCHAR2 column called DUMMY that has a value of ‘X’. It is suitable for use in selecting a pseudo column such as SYSDATE or USER.

--29. Explain, the difference between a function, procedure, and package?

The difference is that a function always returns a value or a table. It cannot make change to the database such as executing an INSERT, UPDATE, or DELETE statement. You code the return keyword followed by data type that’s returned by the function. A procedure on the other hand does not have any specific return type and doesn’t return single but multiple values. While a package is a schema object and it us responsible for grouping PL/SQL types, subprograms, and items, which are logically related. It is group of related procedures and functions, together with the cursors and variables they use, stored together in the database for continued use as a unit.

--30. What is the difference between SQL and PL/SQL?

The difference is that SQL is a declaration language and can perform a single operation at a time, while PL/SQL is a procedural language and can execute multiple operations at the time.

--31. What is the difference between row-level triggers and statement-level triggers?

The difference is that row-level triggers execute once for each row in a transaction and are the most common types of triggers; they are often used in data auditing applications. A row-level trigger is identified by the FOR EACH ROW clause in the CREATE TRIGGER command. While statement-level triggers execute once for each transaction. They are not often used for data-related activities; they are normally used to enforced additional security measures on the types of transactions that may be performed on a table. Statement-level triggers are the default types of triggers created and are identified by omitting the FOR EACH ROW clause in the CREATE TRIGGER command.

--32. What is the importance of %type and %rowtype data types in PL/SQL?

The %type helps us to declare a constant, variable, etc a same type as the previously declared variable or column type. It is used for anchoring for example-the variable emp\_manager has same data type as the column manager\_name in the table employee.

A %rowtype is an attribute that helps us to declare a record that represents a row in the table. The fields of the row have same name and data types as column in the view. Example- dep\_rec dept%ROWTYPE declares a record that can store an entire row for DEPT table.

--33. What is the difference between rollback and rollback to statements in PL/SQL?

The difference is that a rollback statement makes a transaction completely undone where all the blocks are released while a rollback to statement makes a transaction undone but still a SAVEPOINT. This means that the transaction remains live and active even after the command is implemented.

--34.What are the mandatory keywords in a PL/SQL program block?

--They are DECLARE, BEGIN, EXCEPTION, and END

--35.What is a bind variable and how is it used?

--A bling variable is an SQL feature that that lets you turn part of your query into a parameter. This parameter can be provided into the query when you run it, and the query is constructed and is executed. Bind variables which is often called bind parameter or query parameters, are often used in WHERE clauses to filter data.

A bind variable can be used instead of putting the required-value into the query or writing separate queries for each different value or concatenating strings.