A close-up of a document

Description automatically generated

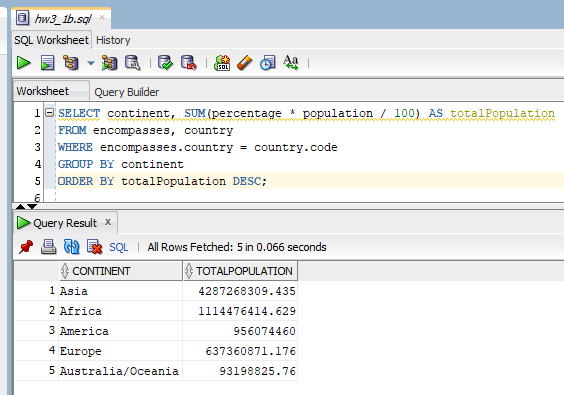
Question 1

a)

1) As stated [here](https://dbcode.io/docs/sql/deferrable): “DEFERRABLE is a constraint attribute in SQL that specifies that the enforcement of the constraint can be postponed until the end of a transaction. This enables a transaction to insert data [in]to a database even though it initially violates a constraint, provided that the violation is corrected before the transaction commits. When a constraint is declared as DEFERRABLE, a temporary violation is allowed in a specific scope of the transaction, known as the deferrable time.”

“By further specifying INITIALLY DEFERRED, it enforces that the constraint checking will always be deferred till the transaction’s end. The data insertion doesn’t violate the foreign key constraint as the check is performed after both insertions are executed.” This is contrasted with the INITIALLY IMMEDIATE option which causes constraint validation to happen immediately unless deferred is specifically requested.

2) The problem arises from adding foreign key constraints. Specifically in this instance the foreign keys in the tables create circular references to each other. If these constraints were not deferable, we would attempt to add a row to one of these tables, only to find that a required entry in another table, as specified by the foreign keys, does not exist, as we haven’t had a chance to create it yet. Making these constraints deferable allows us to enter the data into each of the tables and have the foreign key constraints checked at the end of the transaction, once we’ve had a chance to enter in all of the relevant data.

b)   
1)  


2)  
A screenshot of a computer

Description automatically generated

3)

A screenshot of a computer

Description automatically generated

4)

A computer code with text

Description automatically generated

5)

A screenshot of a computer

Description automatically generated

6)

A screenshot of a computer

Description automatically generated

7)

A screenshot of a computer

Description automatically generated

8)

A screenshot of a computer

Description automatically generated

9)

A screenshot of a computer

Description automatically generated

10)

A screenshot of a computer program

Description automatically generated

11)

A screenshot of a computer

Description automatically generated

12)

A screenshot of a computer

Description automatically generated

13)

A screenshot of a computer

Description automatically generated

14)

A screenshot of a computer

Description automatically generated

15)

A screenshot of a computer

Description automatically generated

Without additional guidance I took the question literally, only looking at those countries with the specific ‘Christian’ religion. In the event we were supposed to look at every denomination of Christian, I would have used an “AND R2.name IN (‘denomination1’, ‘denomination2’, …) construct (plus a whole lot of googling to determine how many of the possible religions are some variant of Christianity).

16)

A screenshot of a computer

Description automatically generated

17)

A screenshot of a computer

Description automatically generated

18)

A screenshot of a computer

Description automatically generated

Question 2

To create persistent queries, I utilized the CREATE VIEW functionality for all of the specified tables (GDPResults, GDPOfCountry, GDPPerCapita, GDPByIndustryPercentage). In order to create the rank order for GDP, GDP per capita, and Industrial GDP Percentage, I utilized the RANK() OVER(ORDER BY [insert column name here] DESC). The RANK function allows ties, and it wasn’t specified whether we should allow this or not. In the event that we need to create a strict 1-N ranking, the DENSE\_RANK function needs to be used. The final item of note in the creation of the GDPResults view is that the rounding to two decimal places was done via the ROUND function. This is the result of the creation of the GDPResults view, arbitrarily ordered by GDP per capita for viewing purposes:

A screenshot of a computer

Description automatically generated

The three sub-views that pulled from this data were created in a similar fashion, the only difference is that these views ordered the columns differently, and the data is ordered in the view by the specified parameter. Vertical alignment of the column values was accomplished by converting the numerical value to a string using the TO\_CHAR function, and the result of this was used as input into the LPAD (left pad) function. The results of the three sub views are shown below:

GDPOfCountry

A screenshot of a computer

Description automatically generated

GDPByIndsustryPercentage

A screenshot of a computer

Description automatically generated

GDPPerCapita

A screenshot of a computer

Description automatically generated

Conclusion

Two of the final views, specifically the one focusing on overall GDP and the one focusing on GDP per capita, revealed nothing unexpected. The GDP rankings were dominated by large, mostly western countries. Though I do have to admit that Brazil at rank seven was a surprise to me, but this surprise can be entirely attributed to my lack of background knowledge on Brazil’s economy. The GDP per capita ranking was dominated by small, wealthy nations of two general varieties. The first variety are those countries with access to large amounts of natural resources relative to their small population. This includes countries like Qatar and Norway. The other variety are those countries that are widely considered to be tax havens, such as Monaco, Switzerland, and Bermuda. The third view, however, was surprising. My initial expectations were that the list would be somewhat similar to the GDP rankings, dominated by relatively wealthy, western countries. But I failed to consider the gradual transition from industry to services that countries make as they get wealthier and was surprised that most countries on this list were relatively poor and generally considered to be in the ‘developing countries’ category.

Question 3

A) Not possible; the lowest nested subquery (SELECT \* FROM works WHERE employee.enum = works.enum AND works.pnum = projects.pnum) attempts to join tables (employee, projects) that it has not imported, resulting in an error.

B) Not possible; there is no GROUP BY functionality in relational algebra. Some googling has suggested that there have been attempted to extend the set of relational algebra operations (I saw one potential symbol for GROUP BY. It was very squiggly), but nothing that has been introduced in this course allows this functionality.