CMPU4003 Advanced Databases

Continuous Assessment Project

Deadline Thursday 12th December 2024 @ 23:59

Contents

Setup

- Review the document attached to this assignment, CMPU4003 CA Project Data Description.pdf. It
    - Describes a relational database
    - A partial data warehouse design which is derived from that database
- Download the script MusicComp.SQL and MusicCompDBInserts.SQL(attached to the assignment).
    - Run these against in your PostgreSQL instance.
    - It will create and populate a new schema called MusicCompDB containing the relational database (23, 139 rows of vote data).
- Download the script MusicCompDimDB-Setup.SQL (attached to the assignment).
    - It is setup to create a schema MusicCompDImDB which implements the partial data warehouse design.
        - Amend this script so that it will create a schema named with your student number.
        - Run this against in your PostgreSQL instance.
        - It will create a schema containing:
            - The dimension tables for the partial data warehouse (and it will populate these).
            - A basic fact table is included with the relationships established between the dimension tables (this will not be populated and may require changes).

Tasks

Task One – PostgreSQL Data Warehouse

1. You need to decide what changes are necessary to the design of the data warehouse to make answering the following three questions more straightforward when using the data warehouse:

   Q1. For competitions in year 2021 and 2023, for each edition of the programme, what is the total votes cast using each voting mode by viewers of each category (audience or jury) and age group in each county for each participant? There is no need to differentiate between audience and jury.

   Q2. For each county, what is the total number of votes received by each participant in the 2021 and 2023 edition of the programme from audience viewers in the same county? (i.e. viewers voting for participants who come from the same county as the viewer).
   a. Include the county name in the output.

   Q3. For the 2021 and 2023 edition of the programme respectively, for each county, what was the total income earned from audience viewers in that county for each voting mode?
   - Include the county names and the year in the output.

   The TV Company charges audience voters to cast their votes. The charge varies depending on the voting mode:
   a. From 2013 to 2021 the charges were:
      - 20c for votes cast by Facebook and Instagram
      - 50c for votes cast using the TV company's website and by Phone.
   b. From 2022 to 2024 the charges were:
      - 50c for votes cast by Facebook and Instagram
      - 1€ for votes cast using the TV company's website and by Phone.
   c. Jury voters are not charged for casting their votes.
   - Hint: Consider the charges for voting. Would this influence the fact design?

2. Write the ETL code needed to make those changes and populate the (changed) fact table (and any other tables you consider that you need) with data from the original database.
   - Include code to undertake any data transformations needed.
   - You can use either a SQL or Python script (or a combination) to achieve this task.

3. Write the queries needed to answer the three questions posed in point 2 above using the data warehouse.
   - To help you understand what is required, download and review the script MusicCompQueries.SQL (attached to the assignment). It includes the queries to answer the three questions posed using the original relational model (MusicCompDB).
   - You can use either a SQL or Python script (or a combination) to achieve this task.

Task Two – Linked Documents CouchDB

1. Design a partitioned document model for CouchDB using a linked document approach to achieve the following:
    - You are concerned only with participants for the competition in 2021 and 2023 (which should form the partitions – one for 2021 and one for 2023).
    - You require two different types of documents to achieve the following
        - For each participant in each edition create one document that stores the total votes achieved by that participant for each vote mode and the amount earned from these votes.
            - Document ids must include the participant's name in the document id plus an indicator of the vote mode e.g. F for Facebook, I for Instagram, W for Web, P for phone.
            - i.e. if a participant achieved votes by Facebook, Instagram, Web and Phone there would be 4 vote documents created.
        - For each participant in each edition create one document including the name of the participant and the edition year plus a list of the vote documents that reflect the votes achieved by each mode for that participant in that year.
            - Document ids must include the name of the participant in document id.
            - This document must include the document ids of the associated vote documents.
    - Write the ETL code to populate a CouchDB database based on this model using data from your PostgreSQL data warehouse.
        - Your database should be named with your student number.
        - The database should have two partitions: one for 2021 and one for 2023.
            - Each partition will store both fact and vote document types.
        - Port the necessary data from your PostgreSQL database.
            - You can use either a combination of SQL and a shell script or Python (or a combination) to achieve this.
    - Implement a global query to return for each fact, the participant's name, year of the competition and the ids of all the associated vote documents.
    - Create a JSON export of your CouchDB database which can be used load data into a Couchdb databases using _bulk_docs.
    - An example database is shown on the next page:

2021:Aaron Baldwin-9146d479-ef4e-4b3e-8b4d-35325f6e5cf3	{ "pname": "Aaron Baldwin", "year": 2021, "votes": ["2021:Aaron Baldwin-I-92bdfc44-4...	fact
2021:Aaron Baldwin-F-be4cb2ff-41fb-47dd-aa24-f063ee0840ee	{ "vote_mode": "Facebook", "total_votes": 51 }	vote
2021:Aaron Baldwin-I-92bdfc44-4a65-40c2-aa40-39037abba548	{ "vote_mode": "Instagram", "total_votes": 46 }	vote
2021:Aaron Baldwin-P-c395cd7f-59d4-492f-a45a-4740b070e222	{ "vote_mode": "Phone", "total_votes": 32 }	vote
2021:Aaron Baldwin-W-e35472c6-c8ec-4e19-ab76-7d559e9e5e4d	{ "vote_mode": "Web", "total_votes": 34 }	vote
2021:Amber Mcdowell-54d78630-9080-4e3a-8b94-ebe1c7e55d1c	{ "pname": "Amber Mcdowell", "year": 2021, "votes": ["2021:Amber Mcdowell-P-917a2...	fact
2021:Amber Mcdowell-F-b66f210a-8886-4e0b-bc49-a5f5d4317674	{ "vote_mode": "Facebook", "total_votes": 42 }	vote
2021:Amber Mcdowell-I-f6ae5775-c793-438c-8897-cc643d0f924f	{ "vote_mode": "Instagram", "total_votes": 46 }	vote
2021:Amber Mcdowell-P-917a2c5d-2c87-4576-ae0f-7bc33e11ebed	{ "vote_mode": "Phone", "total_votes": 31 }	vote
2021:Amber Mcdowell-W-bdc1a0fd-b57e-4a52-923b-f1c632d95566	{ "vote_mode": "Web", "total_votes": 42 }	vote

Task Three – Column Store using a Collection Datatype in Cassandra

1. You can use either a combination of SQL, CQL and JSON or Python (or a combination of all) to achieve this.
2. Design a column store model for your vote data which uses a collection datatype to store the total votes achieved by each participant for each voting mode for participants in the 2021 and 2023 edition.
    - You will have one row of data for each participant in each year.
3. Write the ETL code to create and populate this table.
    - Your keyspace should be named with your student number.
4. Implement a query against this data to demonstrate the table design and data population.

Submission

A single archive (.zip, .rar etc) named with your student number e.g. D123456.zip containing:

Aspect	Requirement	File Naming
PostgreSQL	1. Appropriately commented code to amend the data warehouse, explaining your design choices. 2. Appropriately commented code to populate the fact table (and any other tables) from the original MusicCompDB data. 3. Appropriately commented code to implement the queries to answer the questions posed using the data warehouse.	Either an SQL or Python script named with your student number as follows: • D123456Postgres.sql OR • D123456Postgres.py PLUS Any additional files e.g. JSON data needed for this to work.
CouchDB	1. Appropriately commented code needed to create and populate an equivalent CouchDB database, explaining your design choices. 2. Appropriately commented code needed to implement the global query. 3. A JSON export of your database (including your design document(s)) that can be loaded using _bulk_docs.	You can submit your code as either shell curl script or Python script named with your student number as follows: • D1233456.SQL Plus D123456Couch.sh or D123456Couch.bash OR • D123456Couch.py PLUS • Your data export named with your student number e.g. D123456CouchData.json. PLUS Any additional files e.g. JSON data needed for this to work.
Cassandra	1. Appropriately commented code needed to create your keyspace and create and populate a table that uses a collection data type, explaining your design choices. 2. Appropriately commented code for your query to demonstrate the table design.	You can submit code as either a CQL script or a Python script • D123456.sql Plus D123456Postgres.cql OR • D123456Postgres.py PLUS Any additional files e.g. JSON data needed for this to work.

Marking Scheme

This part of the assessment will be marked out of 100 but weighted to 20% of the module marks.

	Aspect	**Marks**
PostgreSQL	Completion of data warehouse design	10
	Population of data warehouse (fact table plus any additional tables required)	10
	Implementation of Queries to answer required questions using the data warehouse (Three required – 3 x 5 marks)	15
CouchDB	Design of CouchDB Linked Document Model	10
	Creation and population of database	10
	Implementation of a global query	10
Cassandra	Design of Column Store with Collection datatype	10
	Creation of keyspace and creation and population of the table	10
	Query to illustrate table design and population	5
ALL	Appropriate levels of comments explaining and justifying design choices for all aspects	10
	Total Marks (100)	**100**