THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

INFS2200/7903 PROJECT ASSIGNMENT 2

Semester 2 2023

Total Marks:	70 marks (15% of the course)
Due Date:	4:00PM Friday 27-October-2023
What to Submit:	SQL script file + short report
Where to Submit:	Electronic submission via Blackboard

The goal of the project assignments is to gain practical experience in applying several database management concepts and techniques using Oracle DBMS. In particular, this assignment mainly focuses on improving database efficiency with views, indexing, and query planning.

Your main task is to first populate your database with appropriate data, then design, implement, and test the appropriate queries to perform the tasks explained in the next sections.

You must work on this project **individually**. Academic integrity policies apply. Please refer to *3.60.04 Student Integrity and Misconduct* of the University Policy for more information.

Roadmap: Section A describes the database schema for the assignment and provides instructions on downloading the script file needed to create and populate the database. Section B describes the tasks to be completed for this assignment. Finally, Section C explains the submission guidelines and marking scheme.

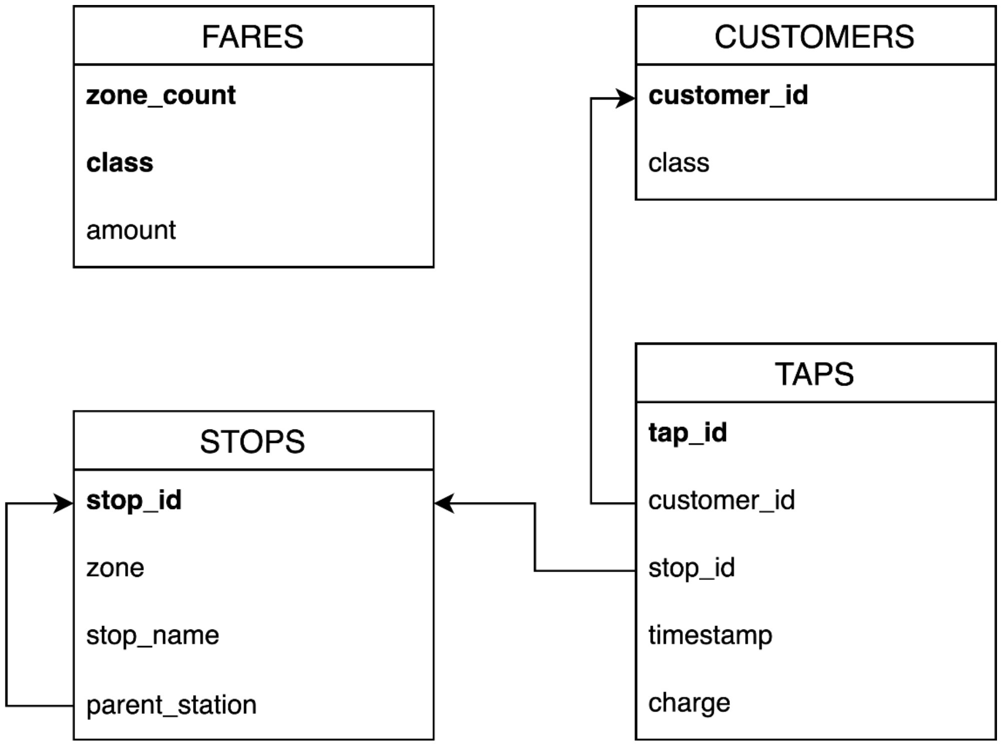
Enjoy the project!

Section A — The TRAVEL database

The TRAVEL database captures information about public transport users' journeys and their fares. The database includes four tables: CUSTOMERS, FARES, STOPS and TAPS. Users travel around the network, "tapping on" with their fare card when they enter a station or vehicle, and "tapping off" when they exit a station or vehicle.

- **CUSTOMERS** stores each customer's fare class (e.g., 'Adult' or 'Concession') and their fare card ID.
- **FARES** stores the price in cents, for each combination of class and the number of zones travelled.
- **STOPS** stores information about each stop on the network, particularly its name and fare zone.
  - Some stops are grouped together via the parent_station relation.
- **TAPS** stores the travel history of each customer, in the form of the stops they have tapped on or off at, and the fares they were charged along the way.

All fares are calculated with the "basic" method described in Assignment 1: a tap-on is free, and a tap-off incurs a (positive) charge determined by the inclusive count of zones travelled as well as the customer's fare class.



Please go to Blackboard and download the supplementary script file for this assignment, **TravelDB.sql**

Note: *this file has been updated since Assignment 1. The first few lines of the updated file will contain a comment confirming it is for Assignment 2.*

Section B — Assignment Tasks

You will first need to execute the script file `TravelDB.sql` to create and populate your database before working on the following tasks. Wait until you see the message "`DONE! All data has been inserted.`" This may take a couple of minutes. Tables and any triggers or sequences from Assignment 1 will also be dropped.

Tip: *For faster reloading, you can export the database to a `.dmp` file.*

Task 1 – Views

1. **Write** an SQL statement to find the top ten (10) stations by revenue. A station is defined as the stops sharing a `parent_station`, and is named by the `stop_name` of the entry referred to by `parent_station`. Revenue for a trip is allocated to the tap-off stop. Your query should display the names of the stations and the corresponding revenues, largest to smallest.

2. **Write** an SQL statement to refine your Task 1.1 query to include only revenue from users with the 'Concession' fare class who have travelled more than one zone. Your query should display the names of the stations and the corresponding revenues.

3. **Write** an SQL statement to create a (virtual) view `V_STATION_BOARDINGS` which shows boardings (tap-ons) at every station, aggregated by days of the week and by fare class. Your view should have a column for station name, fare class, day of the week and boardings; and be sorted by the station name, day of the week (Sunday or Monday first) and fare class.
Hints: you may find the following document from Oracle useful for task 1.3.
https://docs.oracle.com/cd/B28359_01/server.111/b28286/functions001.htm#i88891
You may also wish to research the `NLS_TERRITORY` parameter.

4. **Write** an SQL statement to create a materialized view `MV_STATION_BOARDINGS` that lists the same information as in Task 1.3.

5. Execute the following two SQL statements. **Report** both their query execution time and query execution plan.
```
SELECT * FROM V_STATION_BOARDINGS;
SELECT * FROM MV_STATION_BOARDINGS;
```
Question: Did the materialized view improve the query efficiency? Explain your answer. *Hint: Refer to both the elapsed time, and the cost in the execution plan.*

Task 2 – Indexes

1. The Marketing department has asked you to identify any stops which are named similarly to characters from a recently-released movie. These stop names should contain a word starting with either of the following:

    - "A capital B, then zero or more vowels in any combination, then a single or double R, then zero or more vowels in any combination again, then a B."
    - "A capital K, then one or more vowels in any combination, then an N."

    (For this purpose, a vowel is any of A, E, I, O, U or Y, and letters not explicitly described as capitals may be uppercase or lowercase.)
    Write an SQL statement to find every stop matching the criteria, sorted alphabetically by `stop_name`.

Hint: A "regex" is an expression compactly describing patterns in text.
The following documentation may be helpful:

- https://docs.oracle.com/cd/B28359_01/server.111/b28286/ap_posix001.htm#SQLRF5542
- https://docs.oracle.com/cd/B28359_01/server.111/b28286/functions136.htm#SQLRF06300

2. In order to potentially speed up the query in Task 2.1, a function-based index could be created on the `STOPS` table. **Write** an SQL statement to create an index `IDX_PINK` that best fits this task.

3. Report the execution time and execution plan of the SQL query you wrote in Task 2.1 before and after creating the index in Task 2.2.
   Question: Did the index improve the query efficiency? Explain your answer.
   Hint: You should look at both the elapsed time and the cost in the query execution plan.

4. The Transport department would like to investigate their most consistent users. Write an SQL statement to count the number of tap-offs for which there are at least 200 other tap-offs with the same combination of `customer_id, stop_id` and `charge` values.

5. In order to potentially speed up the query in Task 2.4, bitmap indexes could be created on the TAPS table. **Write** the SQL statements to create indexes `BIDX_CUST_ID`, `BIDX_STOP_ID` and `BIDX_CHARGE` that best fit this task.

6. **Report** the execution time and execution plan of the SQL query you wrote in Task 2.4 before and after creating the indexes in Task 2.5.
   Question: Did the indexes improve query efficiency? Explain your answer.
   Hint: You should look at both the elapsed time and the cost in the query execution plan.

   Note: there's at least two ways to do Task 2.4. One is initially slow, but better demonstrates the effect of the bitmap indexes.

Task 3 – Execution Plan

1. A B+ tree index `PK_TAPID` has been generated automatically for the primary key column `tap_id` of the `TAPS` table. Write the SQL statements to answer the following ***questions:***

    a. What is the height of the B+ tree index?

    b. What is the number of leaf blocks in the B+ tree index?

    c. What is the number of block access needed for a full table scan of the tap table?

Hint: *You may find the following documents from Oracle helpful for Task 3.1:*

- https://docs.oracle.com/cd/B28359_01/server.111/b28320/statviews_5119.htm#REFRN29025
- https://docs.oracle.com/cd/B19306_01/server.102/b14237/statviews_4473.htm#REFRN26286
- https://docs.oracle.com/cd/B19306_01/server.102/b14237/statviews_2105.htm#REFRN20286

2. The following SQL statement lists all the taps with a tap_id larger than 100:
`SELECT * FROM TAP WHERE TAP_ID > 100;`
Report the rule-based execution plan chosen by the Oracle optimizer for executing this query.
Question: Explain the query processing steps taking place in this plan.

3. Report the cost-based execution plan chosen by the Oracle optimizer for executing the query in Task 3.2.
Question: Explain the query processing steps taking place in this plan. In your opinion, what are the main differences between the plans you obtained in Task 3.2 and Task 3.3, based on the statistics from Task 3.1 and your calculation?
Hint: You need to estimate the number of block accesses with and without index. You can assume the tap table is sorted by `tap_id`.

4. The following SQL statement lists all the taps with a `tap_id` larger than 73,900:
`SELECT * FROM TAP WHERE TAP_ID > 73900;`
Report the cost-based execution plan chosen by the Oracle optimizer for executing this query.
Question: Explain the query processing steps taking place in this plan. In your opinion, what are the main differences between the plans you obtained in Task 3.3 and Task 3.4, based on the statistics from Task 3.1 and your calculation?

5. The following SQL statement lists all information for the tap with a `tap_id` of 10,000:
`SELECT * FROM TAP WHERE TAP_ID = 10000;`
Report the cost-based execution plan chosen by the Oracle optimizer for executing this query.
Question: Explain the query processing steps taking place in this plan. In your opinion, what are the main differences between the plans you obtained in Task 3.3 and Task 3.5, based on the statistics from Task 3.1 and your calculation?

SECTION C — Deliverables & Marking Scheme

The project is due by **4:00PM, 27 October 2023.** Late submissions will be penalized unless you are approved for an extension (refer to Section 5.3 of the ECP).

You are required to turn in two files (use StudentID to name your files):

1. StudentID.pdf: (replacing StudentID) – Submit on Blackboard via the Turnitin link "Report Submission"
   A report that answers all the questions in Section 2 including all the necessary SQL statements and the screenshots of their outputs.

2. StudentID.sql: (replacing StudentID) – Submit on Blackboard via the upload link "SQL Script Submission"
   A plain-text script file that includes all your SQL statements.

Your **report** file should include the following content:

- Answers to all the **questions** in Section B.

- If you are asked to **write** SQL statements, you need to include those statements in your report.

- After you execute a SQL statement, if Oracle produces any output (e.g. query result, query execution time, query plan, etc), you should also include a screenshot of the output as well.
  Note: Please be sensible when including query output. Any output close to the size of one page can be shown by just including the first 10 lines and the last 10 lines. A report that includes multiple pages of a query output will lose presentation marks. You may find some helpful instructions for formatting query output in Practical 1 or the following Oracle documentation.
  https://docs.oracle.com/cd/A57673_01/DOC/server/doc/SP33/ch4.htm

Your script file must be in plain text (`.sql`) format. You must make sure that your script file can be executed on the ITEE lab computers by the "@" command. The same SQL statements in your script file should also be copied and pasted into your report file (as explained above). Even though the script file does not introduce any new information compared to the report, it is intended to help the marking tutors to quickly check the correctness of your SQL statements before checking the details in your report file.

Your Assignment 2 mark will be halved if you omit one of the script file or the report.

Enjoy the project! Good luck!

Marking Scheme

Task	Marks	Criteria
1.1	5	Write only one SQL query and generate the correct result.
1.2	5	Write only one SQL query and generate the correct result.
1.3	6	View is created with the correct name and semantics. The correctness of the view will be tested by `SELECT * FROM V_STATION_BOARDINGS;`
1.4	4	Materialized view is created with the correct name and semantics. The correctness of the view will be tested by `SELECT * FROM MV_STATION_BOARDINGS;`
1.5	4	Query execution times and query plans are reported for both view and materialized view. Answer the Questions correctly.
2.1	4	Write only one SQL query and generate the correct result.
2.2	2	Function-based index is created with the correct name and semantics
2.3	4	Query execution time and query plans are reported for both before and after index. Answer the Question correctly.
2.4	4	Write only one SQL statement and generate the correct result.
2.5	3	Bitmap indexes are created with the correct name and semantics.
2.6	4	Query execution time and query plans are reported for both before and after index. Answer the Question correctly.
3.1	5	Write the correct SQL statements to generate index statistics and table statistics respectively. Answer the Questions correctly.
3.2	3	Rule-based execution plan is generated. Answer the Question correctly.
3.3	5	Cost-based execution plan is generated. Answer the Question correctly.
3.4	5	Cost-based execution plan is generated. Answer the Question correctly.
3.5	5	Cost-based execution plan is generated. Answer the Question correctly.
Presentation	2	No query result exceeds one page in the screenshot. No tuple is broken into multiple lines in the screenshot.

END OF ASSIGNMENT