

# Department of Computer Science and Engineering Database Management Systems — CMPS 451 Dr. Rehab Duwairi | Fall 2023 Course Team Project

|  |  |  |
| --- | --- | --- |
| ***STUDENT NAME OF SECTION L51*** | ***STUDENT QUID*** | ***STUDENT EMAIL*** |
| Fatma Ali Elbakry Elbadrawy | 201901981 | fe1901981@qu.edu.qa |
| Ferial Mohamed Zakaria Awadalla Marzouk | 201706475 | fm1706475@qu.edu.qa |
| Sara Metwally Ramadan Said | 202004852 | ss2004852@qu.edu.qa |

**Part A.1 — Description of the application**

1. **DBMS software:**
2. Oracle SQL Development was used to create the relations’ tables and the metadata tables. The relations chosen were (*Airplane* and *Airplane\_Type*) from *Airline ERD*.
3. **UI/GUI:**
   1. Eclipse IDE (Java Development) was used to create a mix between an interactive “console” and an interactive window using JSwing.

**Part A.2 — Application folders**

1. **SQL code:**

****

1. **Eclipse Application: (these files were also attached with the submission)**

****

To run the GUI application you need:

1. Eclipse IDE.
2. To import the library .jar files\* “ojdbc11.jar” and “rs2xml.jar” which are attached with the project.
3. Install “Windowbuilder” from “marketplace” in Eclipse.
4. Finally the SQL connection has to be running in SQL.

\* Note: ojdbc11.jar and rs2xml.jar are required for the connection to be established and the “metadata.java” window to get displayed. The IDE jre must also be compatible with the project version and must be updated manually on the device one project is imported into the workspace.

Link to github: https://github.com/CMPS451-Project-Fatma-Ferial-Sara/Course-Project/tree/5a3d159985afaec61dfb5c800f884d0a7fcb8c05/Phase02/Project%20Submission%20Folder

**Part B — Metadata Description:**

We chose to store our metadata as tables inside our database and assumed reasonable values. Our Airline was assumed to have 5000 records for Airplanes and only 15 records for airplane type (since we assumed there shall only be 15 types for this airline.) The following assumptions were taken into consideration:

* Ordering is assumed to be done using the clustering index of each table.
* We have 333 of all types of airplanes except for one type we have 338 airplanes of the same type.
* Some airplanes have the same number of bookings.
* We have equal number of airplane types (5000 /12 = 416), which means 416 of the airplanes have the same number of maximum seats.
* Regarding the booking, 50% of them are fully booked and that gives us 208 seats.
* As for the rest of the airplanes, some of them have the same number of seats booked. For example, Boeing max number of seats is 350 and airbus max seats is 450, but the available seats for both is 50. Then, that’s also a repetition. Considering that, we can say 1000 + 208 = 1208 available seats are distinct.
* The page size is 16KB and average block size is 212bytes. (Record per Block: (16384)/212 bytes = 77)
* The number of blocks for every airplane type: 333/77 = 4.3 blocks.
* Total number of blocks needed in total for unspanned system 5\*15 = 75 blocks for the airplane.
* Airplane type will not follow the unspanned system since we only have 15 records ordered based on the clustering index.
* Fanout (the number or nodes a node can have) was assumed to be 100.

We divided our metadata into 3 separate tables, each with a defined set of information for each metadata type. First, Index Metadata. We decided to group the table name, index name (for the clustering index), index type, secondary index for that table, leaf blocks, and B-Level. Second, Table Metadata. We included table name, index name (of the clustering index), primary key, foreign key, records number, and blocks number for each table. Lastly, Column Metadata. For each column, the table name for that column, column name, MAX and MIN value for numerical attributes, distinctiveness, selectivity, and a brief description.

**Part C — Queries supported:**

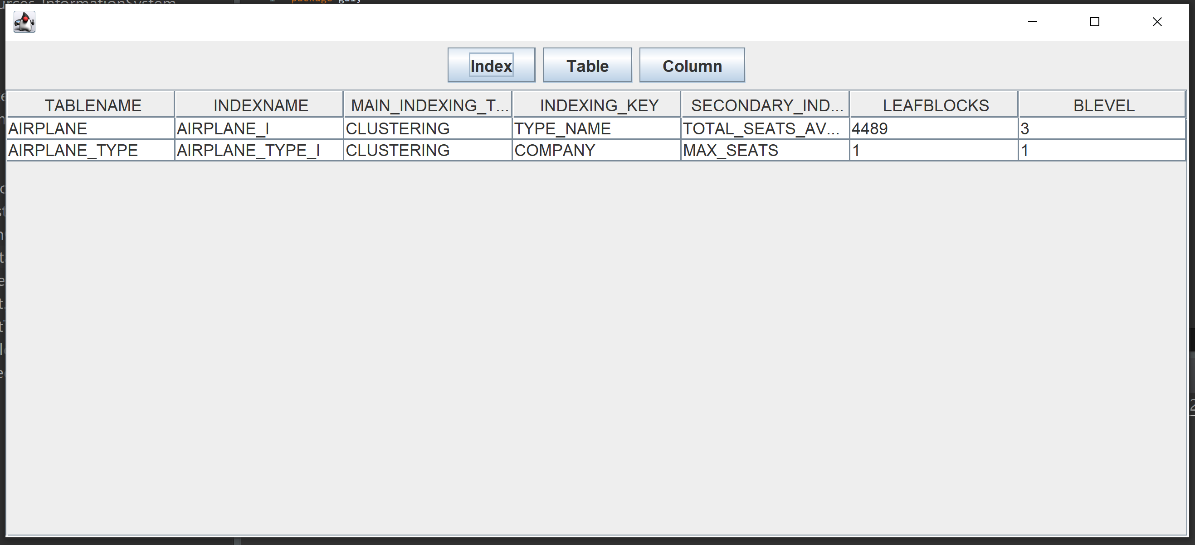
Our program was able to support the following query options:

* Select:
  + A simple select \* from [table name] where [operation] [condition]
  + What is in brackets was given by the user.

A list of all these queries (given examples of conditions):

* + select \* from airplane where airplane\_id = ‘JFK-101’;
  + select \* from airplane where type\_name = ‘Boeing 777’;
  + select \* from airplane where airplane\_id LIKE ‘JFK-101’;
  + select \* from airplane where type\_name LIKE ‘Boeing 777’;
  + select \* from airplane\_type where company = ‘Beoing’;
  + select \* from airplane\_type where type\_name = ‘Boeing 777’;
  + select \* from airplane\_type where company LIKE ‘Boeing’;
  + select \* from airplane\_type where type\_name LIKE ‘Boeing 777’;
  + select \* from airplane where Total\_Number\_of\_Avail\_Seats = 60;
  + select \* from airplane where Total\_Number\_of\_Avail\_Seats < 90;
  + select \* from airplane where Total\_Number\_of\_Avail\_Seats > 10;
  + select \* from airplane where Total\_Number\_of\_Avail\_Seats <= 10;
  + select \* from airplane where Total\_Number\_of\_Avail\_Seats >= 90;
  + select \* from airplane\_type where Max\_Seats = 60;
  + select \* from airplane\_type where Max\_Seats < 90;
  + select \* from airplane\_type where Max\_Seats > 10;
  + select \* from airplane\_type where Max\_Seats <= 10;
  + select \* from airplane\_type where Max\_Seats >= 90;
* Join:
  + Given that we were required to do equi-join, we only needed the primary key and foreign key required for this procedure. The user was not required to fill any fields, they were retrieved from the metadata.

**Part D — Screenshots:**

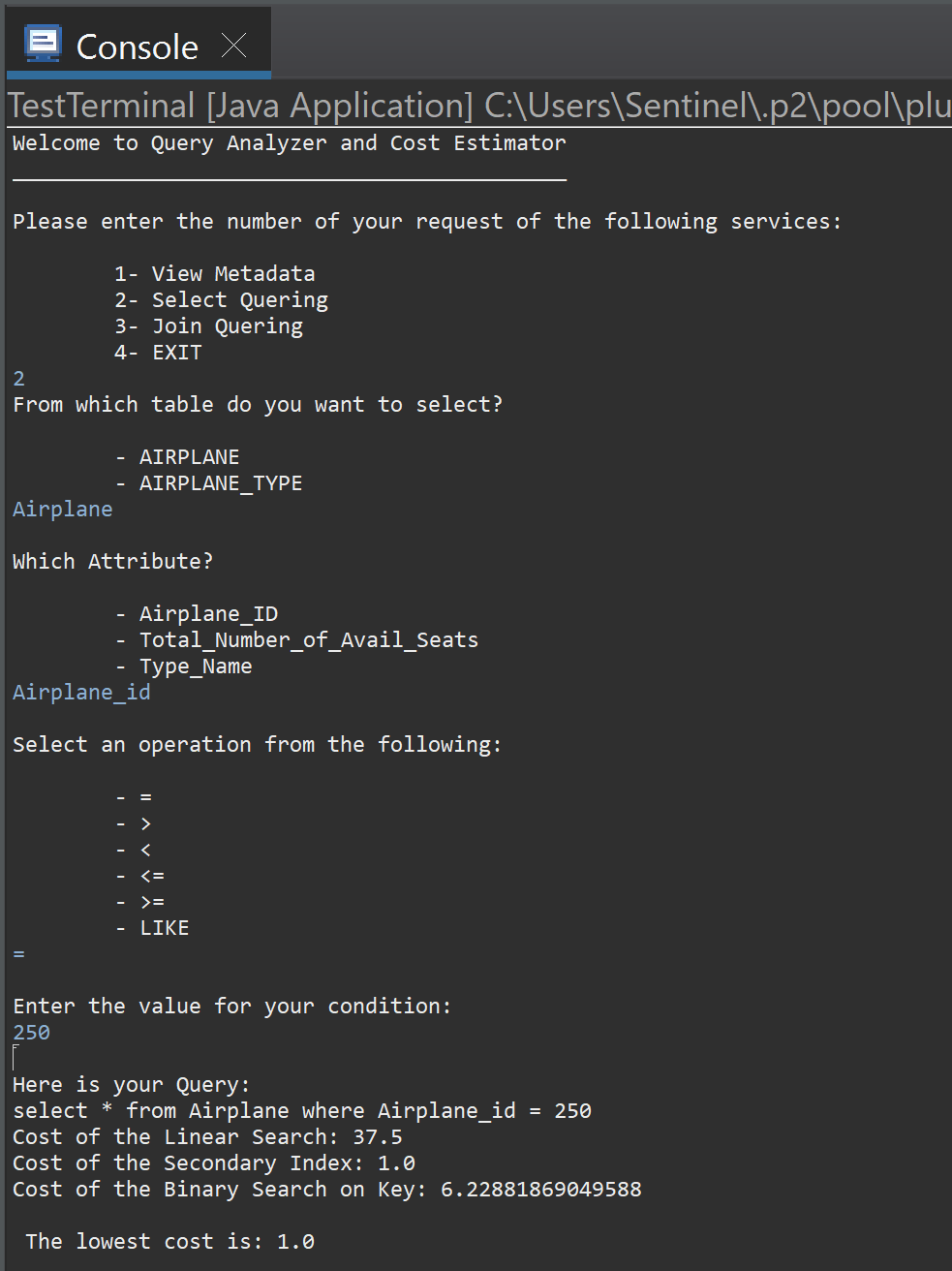
* 1. **Metadata Windows Display**

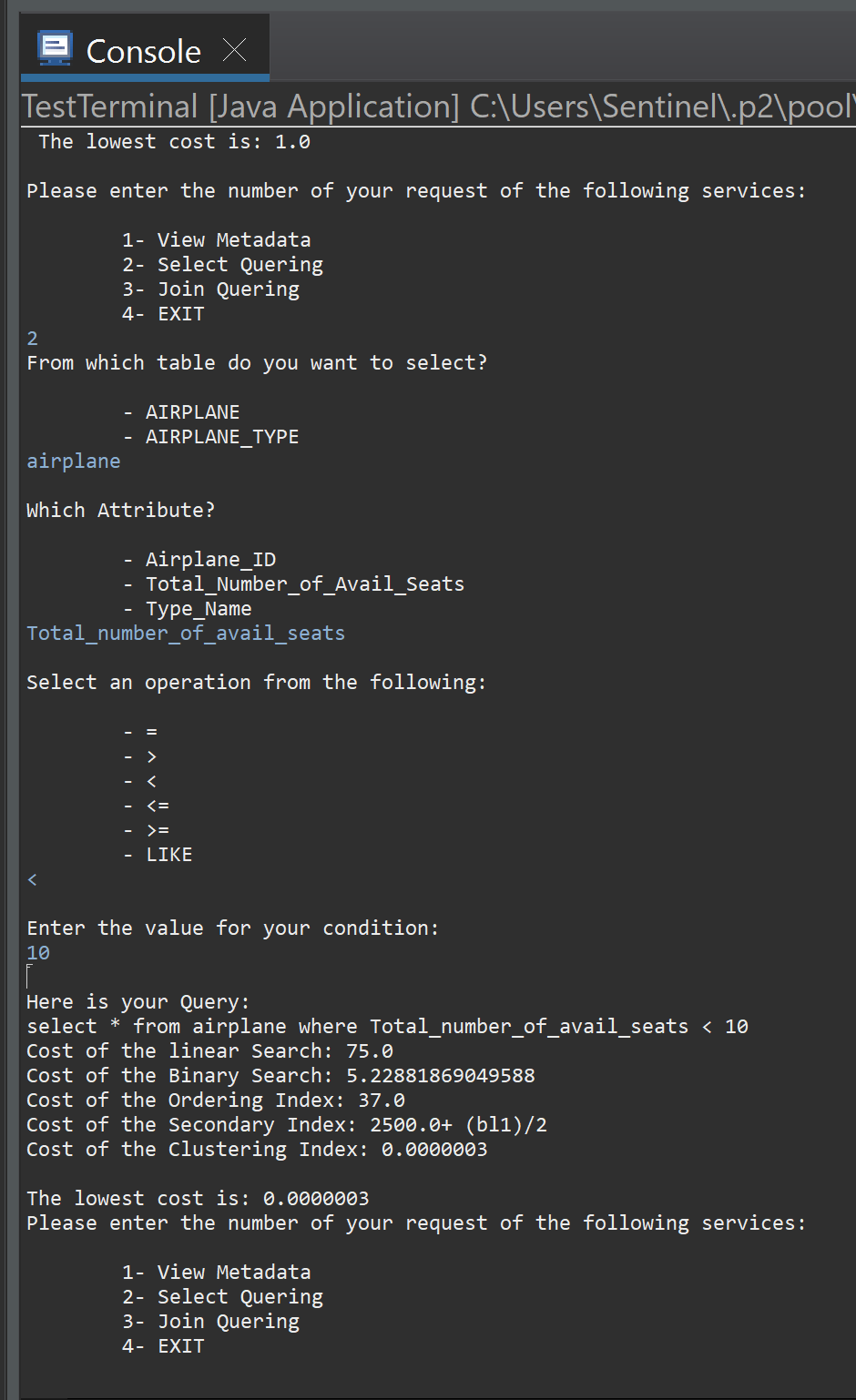
A screenshot of a computer

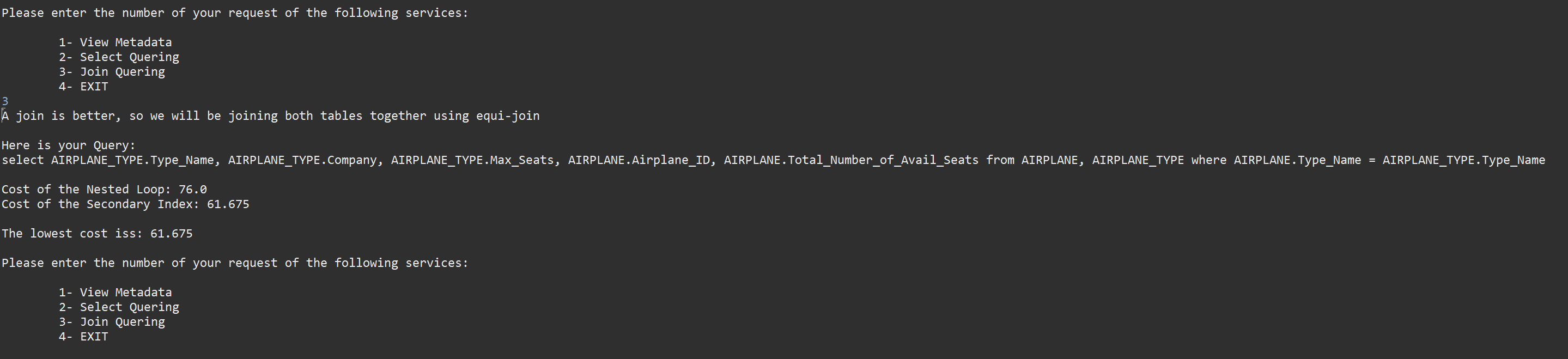
Description automatically generated

A screenshot of a computer

Description automatically generated

* 1. **Select statements:**



* 1. **Equi-Join:**