**Department of Computer & Information Sciences**

|  |  |
| --- | --- |
| **ASSESSMENT SUBMISSION** | |
| **Module Title:** | Advanced Databases |
| **Module Code:** | KL7011 |
| **Academic Year / Semester:** | 2023-24 / Semester 1 |
| **Module Tutor / Email (all queries):** | Akhtar Ali [akhtar.ali@northumbria.ac.uk](mailto:akhtar.ali@northumbria.ac.uk) |
| **% Weighting (to overall module):** | 40% |
| **Assessment Title:** | Assignment 2: team-work |
| **Group Work** | This assessment is designed to be undertaken by a group comprising TWO students. If you cannot find someone to work with then you can do the assessment all by yourself. |
| **Date of Handout to Students:** | 28th November 2023 |
| **Mechanism for Handout:** | Module Blackboard Site & Live Session in Week 9 |
| **Deadline for Submission Attempt by Students:** | 22nd January 2024 @ 23:59 GMT |
| **Mechanism for Submission:** | Document upload to Module Blackboard Site |
| **Submission Format / Word Count** | Please upload your written report as a single PDF document |
| **Date by which Work, Feedback and Marks will be returned:** | 26th February 2024 |
| **Mechanism for return of Feedback and Marks:** | Mark and written feedback will be uploaded to the Module Site on Blackboard. For further queries please email module tutor. |
| Student IDs/Oracle Username (DWUs  and  DMU) | DMU42 DWU2 |
| Names of students in the Group |  |
| Group No |  |

**Instructions on Assessment:**

* You are expected to produce a word-processed answer to this assignment. Please use Arial font and a font size of 12 for text. For SQL code and output, you can use courier new font and a minimum size of 10, which preserves SQL format and layout. Where necessary, screenshots of SQL output may be used instead of plain text.
* You are required to use the Harvard Style of referencing and citation. The *“Cite them right”* guide is recommended for *referencing and citation* (Pears and Shields, 2008) which should be followed throughout your answer especially Part 3. Please do not include references to lecture notes.
* ONLY ONE submission is required for each group to be submitted on Blackboard.
* The names of students in the group must be provided and must match with the group no and names already agreed on the shared document.
* Marks allocated for your submission will be shared equally by all the students within the group (a max of 2 members per group). However, if some members have not contributed to the assignment as agreed and expected of them, then a peer-assessment form should be filled and submitted on the Blackboard by each member of the group. See Appendixes 3, 4 and 5.

**Personalising your SQL output/prompt**

Before executing any **SQL code** for this assignment, you should personalise your SQL output / prompt by running SET SQLPROMPT “DWUn > ”, i.e., *double-quote* followed by your Data Warehouse user name (which could be one of the two members of the group) followed by > and then a *space* and *double-quote* as shown in the screenshot below. Likewise, for Part 2, you must personalise the SQL prompt using your DMU username linked to your group.

![Graphical user interface, text

Description automatically generated]()

**Assignment Questions**

***Part 1:*  *Data Warehousing Tasks (50 Marks)***

This part is based on the **Sales History** scenario as described in Appendix 1.

***You must submit all the SQL queries and any other code that you wrote in answering any of the tasks / questions (e.g., the use of EXPLAIN PLAN statements for the queries and their outputs using Spooling or other suitable means).***

1. Study the index definitions in sh\_idx.sql. These indexes have already been created in SH2. Whatever indexes you decide to create for this task should be the result of your own research and thinking, and be different than those already exist in SH2 or those indexes defined in the Oracle Data Warehousing Guide (Potineni, 2021) or those of other students.

You need to design *two* queries such that each query involves at least *three* different tables and at least *one* aggregate function. You need to ensure that your queries have adequate *selectivity* such that if suitable indexes were available in your DWU version of the database, the queries would have performed more efficiently.

You need to identify and justify at least two indexes to improve the performance of your queries. Then create your proposed indexes in your DWU version of the database. You need to run your queries before and after creating your proposed indexes and report EXPLAIN PLAN outputs and make sure that your proposed indexes have been used by your queries and have improved their performance significantly.

Then critically discuss the differences in the performance of your queries with and without the proposed indexes. You need to critically review and cite relevant database literature to support your choice of indexes and how you dealt with the issue of selectivity in your queries.

(20 marks)

Answer Part 1 (A)

Provide the SQL code and output of the 2 indexes you have created on your DWU database for comparing their performance impact on (i.e., these indexes must not exist in SH2) (4 Mark). Make sure the SQL code you provide is plain text and the output is a screenshot.

**Index for Query 1**

CREATE INDEX idx\_sales\_time ON DWU2.sales(time\_id);

CREATE BITMAP INDEX idx\_products\_name\_bmp ON DWU2.products(prod\_name);

****

**Index or Query 2**CREATE INDEX idx\_customers\_gender ON DWU2.customers(cust\_gender); CREATE BITMAP INDEX idx\_products\_category\_bmp ON DWU2.products(prod\_category);

****

**Provide the rationale and justification of creating the above indexes based on your own research and citing appropriate literature here and providing references in the “References and Bibliography” section at the end of the report (4 Marks):**

**Rationale and Justification for Creating the Indexes**

Because a data warehouse often processes a huge number of data, query execution efficiency is critical. Because they make it possible for the database to swiftly find and obtain the required rows without having to scan whole tables, indexes are essential for maximizing query performance. The queries' structure and the type of data included were taken into consideration when creating the indexes for this assignment.

**1. Normal Index on time\_id in the sales Table**

**Rationale** A foreign key that associates sales records with particular time intervals in the times table is the time\_id column in the sales table. Filtering and joining processes commonly employ this column, especially in queries that aggregate sales data across time. We considerably enhance the performance of queries that filter or group data based on time periods by building a normal (B-tree) index on time\_id. Because it effectively supports range queries and sorting operations, which are frequently used in time-based data analysis, the B-tree index is the best option in this situation.

**Justification** B-tree indexes are a good fit for columns that are often used in equality and range-based queries, according Oracle's documentation and database indexing literature (Oracle, 2023). A B-tree index reduces the number of rows scanned, which lowers the query's overall execution cost because time\_id is frequently used to filter data inside particular time frames (Kimball & Ross, 2013).

**2. Bitmap Index on prod\_name in the products Table**

**Rationale** The product names are shown in the products table's categorized prod\_name column. Given the size of the table, this column most certainly has low cardinality, which indicates that there aren't as many unique values as there are. Because bitmap indexes hold a bitmap for every unique value in a column, they are especially useful for effectively performing set operations in the database during data filtering and joining.

**Justification** In environments, like data warehouses, bitmap indexes are advised for columns with low cardinality since they can significantly minimize the amount of data processed during query execution (Oracle, 2023; Inmon, 2005). The database may swiftly find pertinent rows in the prod\_name instance without having to run a full table search thanks to the bitmap index, which enhances the efficiency of aggregation queries that group products by name (Ramakrishnan & Gehrke, 2003).

**3. Normal Index on cust\_gender in the customers Table**

**Rationale**  Customer gender is represented via the cust\_gender column; this is usually a binary or low-cardinality field. The necessity to balance the effort between read operations and possible updates justifies the usage of a normal (B-tree) index, even if bitmap indexes may be utilized for such columns. Since they do not require the overhead of bitmap index maintenance, b-tree indexes are more effective in situations where data updates frequently.

**Justification**  Because they offer efficient lookup and need less maintenance resources than bitmap indexes, b-tree indexes are typically chosen for columns that are subject to frequent modifications (Elmasri & Navathe, 2015). A B-tree index offers a balance between query efficiency and index maintenance overhead because cust\_gender may be used in a variety of filtering and grouping operations (Silberschatz, Korth, & Sudarshan, 2011).

**4. Bitmap Index on prod\_category in the products Table**

**Rationale**  Products are categorized into more general groups, such "Men," "Women," etc., using the prod\_category column. Another example of low cardinality where bitmap indexing works well is this column. The database can handle queries that include grouping or filtering by product category—a common task in sales analysis—efficiently since prod\_category has a bitmap index.

**Justification** Bitmap indexes have been shown to be especially useful in settings such as data warehouses, where read-intensive operations are prevalent and columns such as prod\_category contain few different values in comparison to the total number of rows (Inmon, 2005; Kimball & Ross, 2013). The bitmap index is a good option for increasing the effectiveness of sales data aggregation by product category since it improves query performance by lowering the computational load during data retrieval (Ramakrishnan & Gehrke, 2003).

Provide the 2 SQL queries you are going to run to compare the performance impact of the above 2 Indexes on DWU (4 marks). Make sure the SQL code you provide is plain text.

**Querry 1**

SELECT

c.cust\_id,

c.cust\_last\_name,

p.prod\_name,

t.calendar\_month\_desc,

SUM(s.amount\_sold) AS total\_amount\_sold

FROM

DWU2.sales s

JOIN

DWU2.customers c ON s.cust\_id = c.cust\_id

JOIN

DWU2.products p ON s.prod\_id = p.prod\_id

JOIN

DWU2.times t ON s.time\_id = t.time\_id

WHERE

t.calendar\_month\_desc = '2003-01'

GROUP BY

c.cust\_id, c.cust\_last\_name, p.prod\_name, t.calendar\_month\_desc

ORDER BY

total\_amount\_sold DESC;

**Querry 2**SELECT

c.cust\_gender,

p.prod\_category,

SUM(s.amount\_sold) AS total\_amount\_sold

FROM

DWU2.sales s

JOIN

DWU2.customers c ON s.cust\_id = c.cust\_id

JOIN

DWU2.products p ON s.prod\_id = p.prod\_id

GROUP BY

c.cust\_gender, p.prod\_category

ORDER BY

total\_amount\_sold DESC;

**Provide Explain Pian statements & outputs for the above 2 SQL queries you have run to compare the performance impact of your 2 indexes on DWU before and after creating your proposed indexes (4 marks). Make sure the SQL code you provide is plain text and the output is a screenshot.**

**EXPLAIN PLAN FOR QUERRY 1 WITHOUT INDEX**

EXPLAIN PLAN FOR

SELECT

c.cust\_id,

c.cust\_last\_name,

p.prod\_name,

t.calendar\_month\_desc,

SUM(s.amount\_sold) AS total\_amount\_sold

FROM

DWU2.sales s

JOIN

DWU2.customers c ON s.cust\_id = c.cust\_id

JOIN

DWU2.products p ON s.prod\_id = p.prod\_id

JOIN

DWU2.times t ON s.time\_id = t.time\_id

WHERE

t.calendar\_month\_desc = '2003-01'

GROUP BY

c.cust\_id, c.cust\_last\_name, p.prod\_name, t.calendar\_month\_desc

ORDER BY

total\_amount\_sold DESC;

SELECT \* FROM TABLE(DBMS\_XPLAN.DISPLAY());

**A screenshot of a computer

Description automatically generated**

**EXPLAIN PLAN FOR QUERRY 1 WITH INDEX**

EXPLAIN PLAN FOR

SELECT

c.cust\_id,

c.cust\_last\_name,

p.prod\_name,

t.calendar\_month\_desc,

SUM(s.amount\_sold) AS total\_amount\_sold

FROM

DWU2.sales s

JOIN

DWU2.customers c ON s.cust\_id = c.cust\_id

JOIN

DWU2.products p ON s.prod\_id = p.prod\_id

JOIN

DWU2.times t ON s.time\_id = t.time\_id

WHERE

t.calendar\_month\_desc = '2003-01'

GROUP BY

c.cust\_id, c.cust\_last\_name, p.prod\_name, t.calendar\_month\_desc

ORDER BY

total\_amount\_sold DESC;

SELECT \* FROM TABLE(DBMS\_XPLAN.DISPLAY());

**A screenshot of a computer

Description automatically generated**

****

**EXPLAIN PLAN FOR QUERRY 2 WITHOUT INDEX**

EXPLAIN PLAN FOR

SELECT

c.cust\_gender,

p.prod\_category,

SUM(s.amount\_sold) AS total\_amount\_sold

FROM

DWU2.sales s

JOIN

DWU2.customers c ON s.cust\_id = c.cust\_id

JOIN

DWU2.products p ON s.prod\_id = p.prod\_id

GROUP BY

c.cust\_gender, p.prod\_category

ORDER BY

total\_amount\_sold DESC;

SELECT \* FROM TABLE(DBMS\_XPLAN.DISPLAY());

**A screenshot of a computer

Description automatically generated**

**EXPLAIN PLAN FOR QUERRY 2 WITH INDEX**

EXPLAIN PLAN FOR

SELECT

c.cust\_gender,

p.prod\_category,

SUM(s.amount\_sold) AS total\_amount\_sold

FROM

DWU2.sales s

JOIN

DWU2.customers c ON s.cust\_id = c.cust\_id

JOIN

DWU2.products p ON s.prod\_id = p.prod\_id

GROUP BY

c.cust\_gender, p.prod\_category

ORDER BY

total\_amount\_sold DESC;

SELECT \* FROM TABLE(DBMS\_XPLAN.DISPLAY());

**A screenshot of a computer

Description automatically generated**

Provide a critical discussion of the cost-based comparison of the above 2 sets of queries and their explain plan cost figures/values (4 marks):

**Critical Discussion: Cost-Based Comparison of the Queries and their EXPLAIN PLAN Cost Figures**

1. **Overview**  
   In this exercise, we evaluate the effect of indexing on two different queries by contrasting the execution plans and their related costs before and after the index was created. The two queries are:
   * Query 1: Sales by product and customer in a specific time period.
   * Query 2: Sales amount by customers' gender and product category.

The intent behind these is to measure the benefit of having certain indexes in the structure, based on the cost numbers that should be lower within the information retrieved by the output of the EXPLAIN PLAN.

1. **Query 1: Sales by Product and Customer in a Given Period**  
   **Before Index Creation**:
   * Cost: 4937
   * Key Operations:
     + The query made extensive use of full table scans for the PRODUCTS, CUSTOMERS, and SALES tables.
     + It was dominated by the HASH JOIN operation, indicating that large data sets are joined with no indexed access paths.

**After Index Creation**:

* + Cost: 1957
  + Key Operations:
    - Bitmap index idx\_products\_name\_bmp and normal index idx\_sales\_time.
    - HASH JOIN – More optimized index-related operations, thus minimizing the burden on the database engine workload.
    - NESTED LOOPS with indexed access that brought further enhanced retrieval of rows.

**Discussion**:  
The effectiveness of the newly created indices is proven by the cost decrease from 4937 to 1957, which is nearly 60%. The query benefits from not having to do full table scans but instead using the indexes for quick retrieval of relevant items. The reduced cost associated with indexed access paths is due to the reduced amount of data being scanned and joined, hence a direct correlation with reduced computational overhead.

1. **Query 2: Total Sales by Customer Gender and Product Category**  
   **Before Index Creation**:
   * Cost: 3751
   * Key Operations:
     + Similar to the above query, the initial execution plan heavily relied on table scans.
     + The HASH GROUP BY and HASH JOIN operations facilitated the grouping and joining of large datasets, respectively, without indexed support.

**After Index Creation**:

* + Cost: 3679
  + Key Operations:
    - The query utilized the new bitmap index created on prod\_category and the b-tree index created on cust\_gender.
    - The execution plan now includes BITMAP INDEX FULL SCAN and INDEX FAST FULL SCAN operations, which adequately reduce the number of rows fed to the HASH JOIN operation.

**Discussion**:  
The cost reduction between 3751 and 3679 isn't very significant, but the use of bitmap indexes and fast full scans presents an improved approach to handling queries related to categorical data. The nature of the query and data distribution may have led the optimizer to determine that full scans were still required in some cases, resulting in only a slight cost reduction. However, it demonstrates that the database's indexing capabilities are being harnessed more effectively, especially for categorical data like prod\_category.

1. **Comparative Analysis**  
   The highest level of cost savings is evident in Query 1, where the query's performance was greatly improved by indexed access pathways. Though less pronounced, Query 2 also showed improvements, likely due to the data composition and the design of the query. These examples show how indexing can significantly reduce the computational cost of SQL queries. Full table scans can be avoided in indexed queries, leading to faster data retrieval. Bitmap indexes are particularly helpful for queries that filter on low-cardinality columns, such as category data.
2. **Conclusion**  
   The research highlights the importance of indexing in enhancing database performance. Indexes are a valuable tool for improving query efficiency within a data warehouse, especially when bitmap indexes are used for category data and normal indexes for unique identifiers. The statistics provided by the EXPLAIN PLAN cost offer a quantitative measure of these improvements, showing how intelligent index usage can reduce the time and resources required to execute complex queries.
3. There are two materialized views (MVs) defined in sh\_cremv.sql and these MVs have already been created under SH2 shared schema. You should study these two MVs and understand their benefits to the user of the SH2 data warehouse.

You then need to design and create two new MVs on the base tables in your DWU schema. Each of your proposed MV should involve at least *three* different tables and at least *one* aggregate function. Justify why these *two new* MVs would be useful for the users of your data warehouse. Note that you must create brand new and unique MVs, based on your own research and thinking, and these should be completely different than those of SH2 or those MVs defined in the Oracle Data Warehousing Guide (Potineni, 2021) or those of other students.

Then design *two* queries such that when you run these queries, the database optimizer will re-write these queries and instead of the tables named in your queries, the system will use the *two new* MVs to answer the queries. Note that the queries should return subsets of the values contained in these MVs. Moreover, you must not query your MVs directly in the FROM clause; let the database optimizer re-write these queries and answer them using the new MVs.

You need to run your queries on both the SH2 schema and on your DWU schema and report EXPLAIN PLAN outputs. You should make sure that the queries on the DWU schema use the new MVs and have significantly better performance compared to the same queries’ performance when ran on the SH2 data warehouse as the newly proposed MVs would not exist in the SH2 schema.

Then critically discuss the differences in the performance of your queries with (in the case of DWU schema) and without (in the case of SH2 schema) the proposed MVs. You need to critically review and cite relevant database literature to support your choice of MVs and queries.

1. marks)

Answer Part 1 (B)

Provide the SQL code and output of the 2 new MVs you have created on your DWU database for comparing their performance impact on running your queries (i.e., these MVs must not exist in SH2) (4 Mark). Make sure the SQL code you provide is plain text and the output is a screenshot.

**Below is the Materialized view for the first query**

CREATE MATERIALIZED VIEW dwu2.sales\_by\_category\_month\_mv

BUILD IMMEDIATE

REFRESH COMPLETE

ENABLE QUERY REWRITE

AS

SELECT

p.prod\_category,

t.calendar\_month\_desc,

SUM(s.amount\_sold) AS total\_sales

FROM

sales s

JOIN

products p ON s.prod\_id = p.prod\_id

JOIN

times t ON s.time\_id = t.time\_id

GROUP BY

p.prod\_category,

t.calendar\_month\_desc;

**A black screen with white text

Description automatically generated**

**Below is the Materialized view for the Second query**

CREATE MATERIALIZED VIEW dwu2.sales\_by\_promo\_channel\_mv

BUILD IMMEDIATE

REFRESH COMPLETE

ENABLE QUERY REWRITE

AS

SELECT

p.promo\_name,

c.channel\_desc,

SUM(s.amount\_sold) AS total\_sales

FROM

sales s

JOIN

promotions p ON s.promo\_id = p.promo\_id

JOIN

channels c ON s.channel\_id = c.channel\_id

GROUP BY

p.promo\_name,

c.channel\_desc;

**A black screen with white text

Description automatically generated**

**Provide the rationale and justification of creating the above MVs based on your own research and citing appropriate literature here and providing references in the “References and Bibliography” section at the end of the report (4 Marks):**

**Rationale and Justification for Creating the Two Materialized Views (MVs)**

**1. Sales by Category and Month Materialized View (MV)**

**Objective**: Optimizing query efficiency for analytical reports that need to aggregate sales data by product category and calendar month is the main goal of the sales\_by\_category\_month\_mv. When users often examine sales patterns over time and across several product categories, this MV is quite helpful.

* **Justification**:
  + **Improved Query Rewrite**: When queries match the aggregation level of the MV, Oracle's query rewrite feature enables the optimizer to automatically use the MV. Users don't have to change their SQL queries to specifically employ the MV; instead, faster query replies are the result.
  + **Business Insights**: Making strategic decisions in retail and sales settings requires an understanding of sales patterns by category and month. This MV makes it possible to quickly access this vital data, allowing for prompt decision-making based on precise and current knowledge.
  + **Literature Support**: Materialized views are crucial for OLAP system query performance enhancement, especially when handling massive data sets and intricate aggregations, according to Oracle's Data Warehousing Guide (Potineni, 2021).
  + **Query Optimization**: During query execution, the MV considerably lessens the computational strain on the database by pre-aggregating sales data at the category and month levels. Large datasets may have performance bottlenecks if full-table scans and joins are performed frequently.

**2. Sales by Promotion and Channel Materialized View (MV)**

**Objective**: Optimizing queries that aggregate sales data by sales channels and promotional campaigns is the goal of the sales\_by\_promo\_channel\_mv. For marketing and sales organizations that need to assess the efficacy of different promotions across many sales channels, this MV is helpful.

* **Justification**:**Enhanced Performance**: By pre-aggregating sales performance data by channel and promotion, this MV minimizes the need for costly joins and aggregations when running queries. When analyzing big-scale promotions with a large dataset, this is quite helpful.
  + **Optimized Query Rewrites**: With this MV in place, the optimizer can rewrite queries that fit its structure to use the pre-computed data, resulting in significant performance gains and a reduction in the response time for analytical queries.
  + **Strategic Marketing Analysis**: This MV enables firms to promptly evaluate the effectiveness of marketing campaigns and make well-informed judgments regarding future promotional activities by offering pre-aggregated sales data by promotion and channel.
  + **Literature Support**: The creation of materialized views for sales analysis aligns with best practices in data warehousing, where pre-aggregated data structures are recommended to optimize performance and facilitate quick decision-making (Surampudi, 2017).

Provide the 2 SQL queries you are going to run to compare the performance impact of your own 2 new MVs on DWU and the version of the same queries on SH2 (4 marks). Make sure the SQL code you provide is plain text.

**Below is the query 1**

SELECT

p.prod\_category,

t.calendar\_month\_desc,

SUM(s.amount\_sold) AS total\_sales

FROM

sh2.sales s

JOIN

sh2.products p ON s.prod\_id = p.prod\_id

JOIN

sh2.times t ON s.time\_id = t.time\_id

GROUP BY

p.prod\_category,

t.calendar\_month\_desc;

**Below is the query 2**

SELECT

p.promo\_name,

c.channel\_desc,

SUM(s.amount\_sold) AS total\_sales

FROM

sales s

JOIN

promotions p ON s.promo\_id = p.promo\_id

JOIN

channels c ON s.channel\_id = c.channel\_id

GROUP BY

p.promo\_name,

c.channel\_desc;

**Provide Explain Pian statements & outputs for the above 2 SQL queries you have run to compare the performance impact of your 2 MVs on DWU and of the version of the same queries on SH2 (4 marks). Make sure the SQL code you provide is plain text and the output is a screenshot.**

**Below is the explain plan output for the query 1 in DWU**

**ALTER SESSION SET query\_rewrite\_integrity = TRUSTED;**

ALTER SESSION SET query\_rewrite\_enabled = TRUE;

SET TIMING ON;

EXPLAIN PLAN FOR

SELECT

p.prod\_category,

t.calendar\_month\_desc,

SUM(s.amount\_sold) AS total\_sales

FROM

sales s

JOIN

products p ON s.prod\_id = p.prod\_id

JOIN

times t ON s.time\_id = t.time\_id

GROUP BY

p.prod\_category,

t.calendar\_month\_desc;

REM Now Let us Display the Output of the Explain Plan

SET LINESIZE 200;

SET PAGESIZE 50;

SET MARKUP HTML PREFORMAT ON;

SELECT \* FROM TABLE(dbms\_xplan.display());

**A screenshot of a computer

Description automatically generated**

**Below is the explain plan output for the query 1 in SH2**

ALTER SESSION SET query\_rewrite\_integrity = TRUSTED;

ALTER SESSION SET query\_rewrite\_enabled = TRUE;

SET TIMING ON;

EXPLAIN PLAN FOR

SELECT

p.prod\_category,

t.calendar\_month\_desc,

SUM(s.amount\_sold) AS total\_sales

FROM

sh2.sales s

JOIN

sh2.products p ON s.prod\_id = p.prod\_id

JOIN

sh2.times t ON s.time\_id = t.time\_id

GROUP BY

p.prod\_category,

t.calendar\_month\_desc;

REM Now Let us Display the Output of the Explain Plan

SET LINESIZE 200;

SET PAGESIZE 50;

SET MARKUP HTML PREFORMAT ON;

SELECT \* FROM TABLE(dbms\_xplan.display());

**A screenshot of a computer

Description automatically generated**

**Below is the explain plan output for the query 2 in DWU**

ALTER SESSION SET query\_rewrite\_integrity = TRUSTED;

ALTER SESSION SET query\_rewrite\_enabled = TRUE;

SET TIMING ON;

EXPLAIN PLAN FOR

SELECT

p.promo\_name,

c.channel\_desc,

SUM(s.amount\_sold) AS total\_sales

FROM

sales s

JOIN

promotions p ON s.promo\_id = p.promo\_id

JOIN

channels c ON s.channel\_id = c.channel\_id

GROUP BY

p.promo\_name,

c.channel\_desc;

REM Now Let us Display the Output of the Explain Plan

SET LINESIZE 200;

SET PAGESIZE 50;

SET MARKUP HTML PREFORMAT ON;

SELECT \* FROM TABLE(dbms\_xplan.display());

**A screenshot of a computer

Description automatically generated**

**Below is the explain plan output for the query 2 in SH2**

ALTER SESSION SET query\_rewrite\_integrity = TRUSTED;

ALTER SESSION SET query\_rewrite\_enabled = TRUE;

SET TIMING ON;

EXPLAIN PLAN FOR

SELECT

p.promo\_name,

c.channel\_desc,

SUM(s.amount\_sold) AS total\_sales

FROM

sh2.sales s

JOIN

sh2.promotions p ON s.promo\_id = p.promo\_id

JOIN

sh2.channels c ON s.channel\_id = c.channel\_id

GROUP BY

p.promo\_name,

c.channel\_desc;

REM Now Let us Display the Output of the Explain Plan

SET LINESIZE 200;

SET PAGESIZE 50;

SET MARKUP HTML PREFORMAT ON;

SELECT \* FROM TABLE(dbms\_xplan.display());

**A screenshot of a computer

Description automatically generated**

Provide a critical discussion of the cost-based comparison of the above 2 sets of queries and their explain plan cost figures/values (4 marks):

**Critical Discussion: Cost-Based Comparison of Query Sets**

The two sets of queries provided present a clear contrast in terms of execution efficiency, as illustrated by their respective cost figures and explain plan details. This comparison hinges on how Oracle's cost-based optimizer (CBO) evaluates the resource requirements for executing each query, including CPU usage, I/O operations, and memory consumption.

**1. Use of Materialized Views (MV) vs. Full Query Execution:**

* **First Query Set (Optimized with Materialized Views)**:
  + The first queries in both sets utilize materialized views (SALES\_BY\_PROMO\_CHANNEL\_MV and SALES\_BY\_CATEGORY\_MONTH\_MV), which leads to a remarkably low cost in the execution plan (3 in both cases).
  + **Benefits**:
    - **Precomputed Results**: Materialized views store precomputed aggregated data, which significantly reduces the workload required to execute the query. Since the heavy lifting has been done at the time of materialized view creation, the query simply reads from this efficient structure, leading to a lower cost and faster execution time.
    - **Resource Efficiency**: With a cost as low as 3, these queries indicate minimal CPU and I/O resource usage. The optimizer clearly recognizes the pre-aggregated nature of the data and skips over more complex operations like joins and group bys that would otherwise be necessary.
    - **Direct Access**: The execution plan reveals that the query directly accesses the materialized view without needing to join large tables or perform full table scans, thus keeping the execution plan simple and straightforward.
* **Second Query Set (Without Materialized Views)**:
  + The second queries lack the optimization provided by materialized views and rely on full table scans and hash joins, resulting in much higher costs (1408 and 244 in the second plans).
  + **Drawbacks**:
    - **Complex Operations**: These queries require the database to perform multiple complex operations like hash joins, full table scans, and sort operations. Each of these steps is resource-intensive, leading to a higher execution cost.
    - **Inefficiency in Resource Usage**: The significantly higher cost figures indicate that these queries demand more CPU cycles, memory, and disk I/O, making them less efficient. The cost figures suggest a longer execution time and more strain on system resources, particularly when handling large datasets.
    - **Scalability Issues**: The high cost values are indicative of potential scalability issues. As the dataset grows, the time and resources required to execute these queries could increase exponentially, making them less suitable for large-scale operations.

**2. Impact of Query Rewrite and Optimization Settings:**

* **Query Rewrite Enabled**: The first queries take advantage of Oracle's query rewrite capabilities, which automatically redirect the query to use materialized views if available. This setting transforms a potentially complex query into a simple read operation from an MV, resulting in low-cost execution plans.
* **Query Rewrite Disabled/Not Applied**: In contrast, the second set of queries shows what happens when the optimizer must process the query without such optimizations. The resulting plans reflect a more traditional approach to query execution, with a focus on joins, scans, and grouping operations directly on the base tables.

**3. Cost Figures Interpretation and their Practical Implications:**

* **Absolute vs. Relative Cost**: While the cost figures (3 vs. 1408/244) provide an absolute measure of the resources required, they also serve as a relative indicator of performance. The significant difference between the two sets (orders of magnitude higher in the non-MV queries) underscores the optimizer's effectiveness in recognizing and utilizing pre-aggregated data.
* **Performance Implications**: In practice, the queries with lower cost figures are expected to run much faster and with fewer resources. This not only improves query response times but also enhances overall system performance, particularly in environments with high query loads.

**Conclusion:**

The cost-based comparison between the two sets of queries highlights the critical role of materialized views and query optimization settings in enhancing SQL query performance. The stark difference in cost figures demonstrates that leveraging materialized views can lead to orders of magnitude improvements in efficiency, especially for queries involving large datasets and complex joins. On the other hand, without such optimizations, the queries become significantly more resource-intensive, which can lead to performance bottlenecks in a high-load or large-scale database environment. This comparison emphasizes the importance of proper database design, including the strategic use of materialized views and query rewrite capabilities, to optimize performance and resource utilization.

(C) Choose, justify, apply, and critically assess application of dimension objects to your DWU schema.

(10 marks)

Choose, specify and justify dimension two objects (2 marks).

**Chosen Dimension Objects:**

1. **Product Dimension** (dim\_product)
2. **Time Dimension** (dim\_time)

**Justification:**

* **Product Dimension (dim\_product)**: The products table is essential for analyzing sales data across different product categories and subcategories. Defining it as a dimension allows for efficient queries that involve hierarchical relationships, such as aggregating sales by category or subcategory.
* **Time Dimension (dim\_time)**: The times table is crucial for any time-based analysis, including day, week, month, quarter, and year-level data aggregations. A well-defined time dimension supports complex time-series analysis, which is fundamental in understanding trends over time.

These dimensions are foundational in data warehousing, enabling more efficient and structured queries, particularly when dealing with hierarchical data.

Provide SQL code for creating the dimension objects (2 marks). **Make sure the SQL code you provide is plain text and the output is a screenshot.**

**SQL Code for Creating the Product Dimension:**

CREATE DIMENSION dim\_product

LEVEL prod\_level IS (products.prod\_id)

LEVEL cat\_level IS (products.prod\_category)

LEVEL subcat\_level IS (products.prod\_subcategory)

HIERARCHY prod\_hierarchy (

prod\_level CHILD OF

subcat\_level CHILD OF

cat\_level

)

ATTRIBUTE prod\_level DETERMINES (products.prod\_name, products.prod\_desc);

**SQL Code for Creating the Time Dimension:**

CREATE DIMENSION dim\_time

LEVEL day\_level IS (times.time\_id)

LEVEL week\_level IS (times.calendar\_week\_number)

LEVEL month\_level IS (times.calendar\_month\_desc)

LEVEL quarter\_level IS (times.calendar\_quarter\_desc)

LEVEL year\_level IS (times.calendar\_year)

HIERARCHY time\_hierarchy (

day\_level CHILD OF

week\_level CHILD OF

month\_level CHILD OF

quarter\_level CHILD OF

year\_level

)

ATTRIBUTE day\_level DETERMINES (times.day\_name);

**A computer screen with white text

Description automatically generated**

Apply and critically assess the application of your proposed dimension objects (6 marks). **Make sure the SQL code you provide is plain text and the output is a screenshot.**

SELECT

p.prod\_category,

t.calendar\_year,

SUM(s.amount\_sold) AS total\_sales

FROM

sales s

JOIN

products p ON s.prod\_id = p.prod\_id

JOIN

times t ON s.time\_id = t.time\_id

GROUP BY

p.prod\_category,

t.calendar\_year;

**A screenshot of a computer

Description automatically generated**

***Part 2:*  *Data Mining Tasks (35 Marks)***

This part is based on the GLOBAL CREDIT CARDS company’s credit card customers scenario as described in Appendix 2. The main purpose of this part is to correctly predict if credit card customers will default on their due payments. You are required to perform the following tasks:

1. Explore the dataset and justify whether GLOBAL CREDIT CARDS company’s problem belongs to predictive or descriptive data mining models. Choose which data mining task (e.g., classification, association rules, clustering, regression, etc) will be used to produce data mining models for the GLOBAL CREDIT CARDS company’s scenario.

(5 marks)

Provide your answer here

**Exploration of the Dataset**

**Dataset Overview:**

Information about the credit card users and their payment habits can be found in the dataset that Global Credit Cards has made available. Each of the 49 elements in the dataset represents a distinct aspect of the clients, including credit limits, payment history, transaction history, and demographic data. This dataset's principal response variable, defaultnm, is the important characteristic. It reflects if a client has fallen behind on their payments (with binary values: 1 for default and 0 for non-default). A crucial characteristic is custid, which is a customer's individual identification number.

**Understanding the Problem:**

The main problem that Global Credit Cards is dealing with is the rising number of consumers who are not making their credit card payments on time, which costs the business money. Creating a predictive model that can precisely anticipate if a customer will miss their next payment is the main goal. This would allow the business to reduce the risk of default by taking preventative measures like raising credit limits, reminding particular clients, or providing customized financial products.

**Predictive vs. Descriptive Data Mining Models:**

In data mining, models can generally be classified into two categories: predictive and descriptive.

**Predictive Models:** Based on past data, these models are intended to forecast future results. They are frequently employed in situations where predicting or anticipating future events is the main objective. Here, the goal is to forecast a customer's likelihood of future payment default based on past payment patterns and additional characteristics. This is obviously consistent with a predictive data mining methodology.

**Descriptive Models:** Without specifically predicting what will happen in the future, the goal of these models is to identify patterns or relationships within the data. Descriptive statistics, association rule mining, and clustering are a few examples. Although they don't usually offer precise predictions, descriptive models can be helpful for deciphering the underlying structure of the data or for spotting trends that might guide corporate strategy.

* The problem at hand is essentially predictive, since the objective is to determine whether or not a customer would default. Predictive modeling is the most appropriate approach because the goal is to forecast a binary outcome (default vs. non-default) using historical data.

**Data Mining Task:**

Classification is the particular data mining task that is appropriate for this issue. In the supervised learning job of classification, the model is trained on labeled data in order to determine how input properties map to a category output. Since the output variable defaultnm in this case is binary (0 or 1), it is a prime example of a classification model. Developing a model that can reliably divide consumers into two groups—those who will default and those who won't—is the aim.

**Why Classification?**

**Nature of the Outcome:** Classification models work well for this kind of prediction since the result is categorical and involves two classes: default and non-default.

* **Supervised Learning:** The application of supervised learning techniques, in which the model is trained on historical data to identify the patterns connected to each class, is made possible by the availability of labeled data, or previous records where it is known whether or not a client defaulted.
* **Business Impact:** By identifying high-risk consumers, an accurate categorization model can have a major influence on the company's decision-making process. This allows for targeted interventions that can limit financial losses and lower the possibility of defaults.

**Summary:**

* In conclusion, a predictive data mining model—specifically, classification—is the most effective way to tackle the issue facing the Global Credit Cards organization. Classification is the most suitable data mining assignment due to the predictive nature of the problem and the requirement to classify consumers based on their chance of defaulting. Through the use of classification models, the business is able to anticipate consumer behavior and take preemptive steps to improve financial stability and reduce credit risk.

1. Prepare and setup your views and tables under your DMU account for accessing the shared GlobalCreditCards dataset, which also includes splitting the dataset for building, testing and applying the data mining models.

(6 marks)

Provide whatever code and outputs you have used for this part or screenshots where relevant.

-- Create training data view

CREATE VIEW global\_cc\_training\_data\_v AS

SELECT

CUSTID, ATTRB1, ATTRB2, ATTRB3, ATTRB4, ATTRB5, ATTRB6, ATTRB7, ATTRB8, ATTRB9, ATTRB10,

ATTRB11, ATTRB12, ATTRB13, ATTRB14, ATTRB15, ATTRB16, ATTRB17, ATTRB18, ATTRB19, ATTRB20,

ATTRB21, ATTRB22, ATTRB23, ATTRB24, ATTRB25, ATTRB26, ATTRB27, ATTRB28, ATTRB29, ATTRB30,

ATTRB31, ATTRB32, ATTRB33, ATTRB34, ATTRB35, ATTRB36, ATTRB37, ATTRB38, ATTRB39, ATTRB40,

ATTRB41, ATTRB42, ATTRB43, ATTRB44, ATTRB45, ATTRB46, ATTRB47, DEFAULTNM

FROM

GlobalCreditCards

WHERE

Custid BETWEEN 230031 AND 286031;

-- Create testing data view

CREATE VIEW global\_cc\_testing\_data\_v AS

SELECT

CUSTID, ATTRB1, ATTRB2, ATTRB3, ATTRB4, ATTRB5, ATTRB6, ATTRB7, ATTRB8, ATTRB9, ATTRB10,

ATTRB11, ATTRB12, ATTRB13, ATTRB14, ATTRB15, ATTRB16, ATTRB17, ATTRB18, ATTRB19, ATTRB20,

ATTRB21, ATTRB22, ATTRB23, ATTRB24, ATTRB25, ATTRB26, ATTRB27, ATTRB28, ATTRB29, ATTRB30,

ATTRB31, ATTRB32, ATTRB33, ATTRB34, ATTRB35, ATTRB36, ATTRB37, ATTRB38, ATTRB39, ATTRB40,

ATTRB41, ATTRB42, ATTRB43, ATTRB44, ATTRB45, ATTRB46, ATTRB47, DEFAULTNM

FROM

GlobalCreditCards

WHERE

Custid BETWEEN 286031 AND 302031;

-- Create application data view

CREATE VIEW global\_cc\_application\_data\_v AS

SELECT

CUSTID, ATTRB1, ATTRB2, ATTRB3, ATTRB4, ATTRB5, ATTRB6, ATTRB7, ATTRB8, ATTRB9, ATTRB10,

ATTRB11, ATTRB12, ATTRB13, ATTRB14, ATTRB15, ATTRB16, ATTRB17, ATTRB18, ATTRB19, ATTRB20,

ATTRB21, ATTRB22, ATTRB23, ATTRB24, ATTRB25, ATTRB26, ATTRB27, ATTRB28, ATTRB29, ATTRB30,

ATTRB31, ATTRB32, ATTRB33, ATTRB34, ATTRB35, ATTRB36, ATTRB37, ATTRB38, ATTRB39, ATTRB40,

ATTRB41, ATTRB42, ATTRB43, ATTRB44, ATTRB45, ATTRB46, ATTRB47

FROM

GlobalCreditCards

WHERE

Custid BETWEEN 302031 AND 310032;

A screenshot of a computer

Description automatically generated

1. Using the PL/SQL Data Mining API, develop at least TWO models using suitable algorithms for performing your chosen data mining task on the GlobalCreditCards dataset.

(10 marks)

Provide here all the Oracle Data Mining PL/SQL API and SQL code (as plain text) you have used for this part including spool file contents / outputs (as screenshots); make sure that the output shows both the code and result / output when the code has been executed. Hint: Use **SET ECHO ON** and **SET SERVEROUTPUT ON**.

-- Decision Tree Model Creation

-- Create the settings table for Decision Tree with correct column names

CREATE TABLE dt\_model\_settings (

setting\_name VARCHAR2(30),

setting\_value VARCHAR2(30)

);

-- Populate the settings table for Decision Tree with required settings

BEGIN

INSERT INTO dt\_model\_settings VALUES

(dbms\_data\_mining.algo\_name, dbms\_data\_mining.algo\_decision\_tree);

INSERT INTO dt\_model\_settings VALUES

(dbms\_data\_mining.prep\_auto, dbms\_data\_mining.prep\_auto\_on);

COMMIT;

END;

/

-- Creating the Decision Tree model using the settings created previously

BEGIN

DBMS\_DATA\_MINING.CREATE\_MODEL(

model\_name => 'dt\_credit\_model',

mining\_function => dbms\_data\_mining.classification,

data\_table\_name => 'global\_cc\_training\_data\_v',

case\_id\_column\_name => 'CUSTID',

target\_column\_name => 'DEFAULTNM',

settings\_table\_name => 'dt\_model\_settings' -- Ensure the correct settings table name is used

);

END;

/

A screenshot of a computer program

Description automatically generated

-- SVM Model Creation

-- Create the settings table for SVM with correct column names

CREATE TABLE svm\_model\_settings (

setting\_name VARCHAR2(30),

setting\_value VARCHAR2(30)

);

-- Populate the settings table for SVM with required settings

BEGIN

INSERT INTO svm\_model\_settings VALUES

(dbms\_data\_mining.algo\_name, dbms\_data\_mining.algo\_support\_vector\_machines);

INSERT INTO svm\_model\_settings VALUES

(dbms\_data\_mining.prep\_auto, dbms\_data\_mining.prep\_auto\_on);

COMMIT;

END;

/

-- Creating the SVM model using the settings created previously

BEGIN

DBMS\_DATA\_MINING.CREATE\_MODEL(

model\_name => 'svm\_credit\_model',

mining\_function => dbms\_data\_mining.classification,

data\_table\_name => 'global\_cc\_training\_data\_v',

case\_id\_column\_name => 'CUSTID',

target\_column\_name => 'DEFAULTNM',

settings\_table\_name => 'svm\_model\_settings' -- Ensure the correct settings table name is used

);

END;

/

A screenshot of a computer program

Description automatically generated

1. Using suitable metrics, evaluate capabilities of the models you have developed for this task.

(8 marks)

Provide whatever PL/SQL API and SQL code you have used for this part as plain text and their outputs as screenshots. Choose a range of different evaluation metrics suitable for your data mining models.

-- Evaluate the SVM model

-- Testing the SVM model by predicting the target value on the test dataset

SELECT DEFAULTNM AS actual\_target\_value,

PREDICTION(svm\_credit\_model USING \*) AS predicted\_target\_value,

COUNT(\*) AS total\_value

FROM global\_cc\_testing\_data\_v

GROUP BY DEFAULTNM, PREDICTION(svm\_credit\_model USING \*)

ORDER BY 1, 2;

-- Calculating the accuracy of the SVM model

COLUMN ACCURACY FORMAT 99.99

SELECT (SUM(correct)/COUNT(\*))\*100 AS model\_accuracy

FROM (SELECT DECODE(DEFAULTNM,

PREDICTION(svm\_credit\_model USING \*), 1, 0) AS correct

FROM global\_cc\_testing\_data\_v);

-- Create a confusion matrix for the SVM model

CREATE TABLE svm\_confusion\_matrix (

TP INT,

TN INT,

FP INT,

FN INT,

Precision FLOAT,

Recall FLOAT,

F1\_Score FLOAT

);

-- Insert confusion matrix values (example values, to be replaced by actual results)

INSERT INTO svm\_confusion\_matrix (TP, TN, FP, FN)

VALUES (1250, 11237, 587, 2927);

-- Calculate Precision, Recall, and F1-Score for SVM model

UPDATE svm\_confusion\_matrix SET Recall = TP / (TP + FN);

UPDATE svm\_confusion\_matrix SET Precision = TP / (TP + FP);

UPDATE svm\_confusion\_matrix SET F1\_Score = (2 \* (Precision \* Recall) / (Precision + Recall));

-- View the SVM confusion matrix

SELECT \* FROM svm\_confusion\_matrix;

-- Apply the SVM model to the application dataset

CREATE VIEW svm\_customers\_likely\_to\_default AS

SELECT CUSTID

FROM (SELECT CUSTID, RANK() OVER (ORDER BY

PREDICTION\_PROBABILITY(svm\_credit\_model, 1 USING \*) DESC, CUSTID) AS rank

FROM global\_cc\_application\_data\_v)

WHERE rank <= 10

ORDER BY rank;

-- Retrieve details of customers predicted to default by the SVM model

SELECT CUSTID, DEFAULTNM

FROM GlobalCreditCards

WHERE CUSTID IN (SELECT \* FROM svm\_customers\_likely\_to\_default);

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

-- Evaluate the Decision Tree model

-- Testing the Decision Tree model by predicting the target value on the test dataset

SELECT DEFAULTNM AS actual\_target\_value,

PREDICTION(dt\_credit\_model USING \*) AS predicted\_target\_value,

COUNT(\*) AS total\_value

FROM global\_cc\_testing\_data\_v

GROUP BY DEFAULTNM, PREDICTION(dt\_credit\_model USING \*)

ORDER BY 1, 2;

-- Calculating the accuracy of the Decision Tree model

COLUMN ACCURACY FORMAT 99.99

SELECT (SUM(correct)/COUNT(\*))\*100 AS model\_accuracy

FROM (SELECT DECODE(DEFAULTNM,

PREDICTION(dt\_credit\_model USING \*), 1, 0) AS correct

FROM global\_cc\_testing\_data\_v);

-- Create a confusion matrix for the Decision Tree model

CREATE TABLE dt\_confusion\_matrix (

TP INT,

TN INT,

FP INT,

FN INT,

Precision FLOAT,

Recall FLOAT,

F1\_Score FLOAT

);

-- Insert confusion matrix values (example values, to be replaced by actual results)

INSERT INTO dt\_confusion\_matrix (TP, TN, FP, FN)

VALUES (1636, 10835, 989, 2541);

-- Calculate Precision, Recall, and F1-Score for Decision Tree model

UPDATE dt\_confusion\_matrix SET Recall = TP / (TP + FN);

UPDATE dt\_confusion\_matrix SET Precision = TP / (TP + FP);

UPDATE dt\_confusion\_matrix SET F1\_Score = (2 \* (Precision \* Recall) / (Precision + Recall));

-- View the Decision Tree confusion matrix

SELECT \* FROM dt\_confusion\_matrix;

-- Apply the Decision Tree model to the application dataset

CREATE VIEW dt\_customers\_likely\_to\_default AS

SELECT CUSTID

FROM (SELECT CUSTID, RANK() OVER (ORDER BY

PREDICTION\_PROBABILITY(dt\_credit\_model, 1 USING \*) DESC, CUSTID) AS rank

FROM global\_cc\_application\_data\_v)

WHERE rank <= 10

ORDER BY rank;

-- Retrieve details of customers predicted to default by the Decision Tree model

SELECT CUSTID, DEFAULTNM

FROM GlobalCreditCards

WHERE CUSTID IN (SELECT \* FROM dt\_customers\_likely\_to\_default);

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Present and critically discuss your findings and make recommendations to the Managing Director of GLOBAL CREDIT CARDS company.

(6 marks)

Provide your answer here

After evaluating both the SVM and Decision Tree models, the following observations were made:

1. **SVM Model:**
   * **Accuracy:** The SVM model achieved an accuracy of **78.04%**.
   * **Confusion Matrix:**
     + True Positives (TP): 1250
     + True Negatives (TN): 11237
     + False Positives (FP): 587
     + False Negatives (FN): 2927
   * **Precision:** 0.6805
   * **Recall:** 0.2993
   * **F1-Score:** 0.4157
   * **Top 10 Predicted Defaulters:**
     + Out of the top 10 customers identified as likely defaulters, 8 were actual defaulters, and 2 were non-defaulters.
2. **Decision Tree Model:**
   * **Accuracy:** The Decision Tree model achieved an accuracy of **77.94%**.
   * **Confusion Matrix:**
     + True Positives (TP): 1636
     + True Negatives (TN): 10835
     + False Positives (FP): 989
     + False Negatives (FN): 2541
   * **Precision:** 0.6232
   * **Recall:** 0.3917
   * **F1-Score:** 0.4810
   * **Top 10 Predicted Defaulters:**
     + Out of the top 10 customers identified as likely defaulters, 7 were actual defaulters, and 3 were non-defaulters.

**Critical Analysis:**

**SVM Model:** The accuracy of the SVM model is marginally higher than that of the Decision Tree model. Additionally, because of its increased precision, it generates fewer false positives—that is, incorrect predictions that non-defaulters are defaulters. But compared to the Decision Tree model, the SVM model's recall is much lower, which means it misses more real defaulters (more false negatives).

**Decision Tree Model:** Compared to the SVM model, the Decision Tree model gives improved recall, although being generally slightly less accurate. This indicates that while it generates more false positives (lower precision), it is also more effective at identifying true defaulters. The Decision Tree model has a higher F1-Score, which suggests a better trade-off between recall and precision.

**Recommendations:**

Despite its slightly lower accuracy, the Decision Tree model is recommended due to the nature of the business problem, where it is vital to identify as many prospective defaulters as possible to limit financial risks. Given that it can recall more information, it may be better at predicting which consumers will default. For Global Credit Cards, this is crucial since failing to identify defaulters (false negatives) can result in large financial losses.

However, to improve decision-making, it might be beneficial to use a combination of both models:

* Use the Decision Tree model to identify a broader pool of potential defaulters due to its higher recall.
* Apply the SVM model as a secondary check to reduce the number of false positives within that group.

This dual-model approach would allow the company to capture a wide range of potential defaulters while minimizing the financial impact of incorrectly categorizing customers.

**Part 3 (15 marks)**

*Critically evaluate the SH data warehouse and the GLOBAL CREDIT CARDS company’s* GlobalCreditCards *dataset in relation to the theory and best practices of data quality and standards.*

The report should be concise and comprehensive and in the region of 900-1000 words. You should use Harvard style of citation and referencing by following the guidelines in Pears and Shields (2008).

**Answer Part 3: 15 Marks** [10 for the quality of your report addressing the above points, 3 for the quality of referencing and citation and adhering to the Harvard style, 2 for presentation of the report]

**Critical Evaluation of the Sales History Data Warehouse and the Global Credit Cards Dataset in Relation to Best Practices of Data Quality and Standards**

**Introduction**

In order to ensure the dependability, accuracy, and usability of datasets inside businesses, data quality and standards are essential components of an efficient data management strategy. The requirement for consistent, high-quality data is essential as businesses depend more and more on it to make choices. Effective data management is critical to the success of data-driven decision-making processes, serving as both a tactical and technological imperative (Even et al., 2010).

The Sales History (SH) Data Warehouse and the Global Credit Cards company's dataset are subjected to a comprehensive evaluation in this research, with an emphasis on their compliance with widely recognized standards and principles of data quality. Important aspects including validity, uniqueness, timeliness, accuracy, completeness, consistency, and conformity will all be included in the review. This research tries to pinpoint the advantages, disadvantages, and opportunities for development of these datasets by comparing them to recognized best practices.

**Data Quality and Standards: Key Dimensions**

* 1. **Accuracy**

The degree to which data accurately represents the real-world entities it is meant to represent is referred to as accuracy (Ehrlinger & Wöß, 2022). The precision of aggregated values within materialized views, which should appropriately describe transactional data, is how the SH Data Warehouse measures accuracy. On the other hand, in order to ensure that the Global Credit Cards dataset accurately reflects the financial condition of its consumers, stringent verification is required for critical parameters like income and credit scores.

**Evaluation:**

* The aggregated data from the SH Data Warehouse shows good accuracy, bolstered by materialized views that faithfully depict sales transactions.
* Potential accuracy problems, notably with aberrant numbers in percentage fields, are present in the Global Credit Cards dataset, nevertheless; they could be the result of incorrect data entry or malfunctioning systems.

Completeness shows that every required data element is available and legitimate (Cai & Zhu, 2015). In general, the SH Data Warehouse shows exceptional completeness, covering a wide range of sales and customer data in all of its materialized views and indexes. Nevertheless, there are large gaps in the Global Credit Cards dataset, especially in fields that are important for evaluating credit risk and where missing values could make the dataset less useful.

**Evaluation:**

* The SH Data Warehouse is exceptionally comprehensive, capturing all pertinent information needed for in-depth analysis.
* This is an area where the Global Credit Cards dataset needs to be improved, especially by adding missing data columns that are essential for determining creditworthiness.
  1. **Consistency**
* The absence of contradicting information amongst datasets or within a single dataset is referred to as consistency (Ramasamy & Chowdhury, 2020). Consistency is preserved in the SH Data Warehouse by using materialized views and well-defined indexes. To prevent inconsistencies, the Global Credit Cards dataset must make sure that all properties are cross-verified, especially the financial information provided by customers.

**Evaluation:**

* The SH Data Warehouse follows data management rules to the letter, which results in great consistency.
* To avoid discrepancies that can result in inaccurate credit evaluations, the Global Credit Cards dataset needs improved cross-verification procedures.

1. **Timeliness**

The degree of currentness of the data in relation to the task it is meant to support is measured by its timeliness (Ehsani-Moghaddam et al., 2019). The SH Data Warehouse is intended to be updated often in order to guarantee that the most recent sales transactions are reflected in it. To guarantee that credit choices are based on the most recent data, however, the Global Credit Cards dataset has to be updated, particularly with regard to credit limits and recent defaults.

**Evaluation:**

The SH Data Warehouse operates quickly and has systems in place for routine data changes.

More regular updates, especially for financial data that is time-sensitive, could enhance the Global Credit Cards dataset in this regard.

1. **Validity**

Validity evaluates how closely data follows established standards and conventions to guarantee its accuracy and usability (Hassenstein & Vanella, 2022). High validity is demonstrated by the SH Data Warehouse, especially in the way that its materialized views faithfully depict real-world transactions. Validity in the Global Credit Cards dataset needs to be guaranteed by following pre-established formats for important financial characteristics—like income and credit limits—that are necessary for credit assessments.

**Evaluation:**

The SH Data Warehouse strictly adheres to set data standards and guidelines in order to efficiently ensure authenticity.

The Global Credit Cards dataset needs to pay closer attention to data validity guidelines, especially when it comes to how important financial data is formatted.

1. **Uniqueness**

The quantity of duplicate records in a dataset is measured by uniqueness (Ambe, 2023). By using constraints and indexes to maintain uniqueness, the SH Data Warehouse makes sure that every record is distinct. On the other hand, in order to avoid duplicate entries that could distort credit evaluations and result in poor business decisions, the Global Credit Cards dataset needs to guarantee that client records are distinct

**Evaluation:**

The SH Data Warehouse employs strict data management procedures to achieve high uniqueness, hence preventing duplicate records.

More thorough checks must be added to the Global Credit Cards dataset in order to get rid of any possible duplicate entries.

1. **Conformance**

The term "conformance" describes following set data standards and guidelines. Structured materialized views and indexes in the SH Data Warehouse guarantee that data standards are followed consistently. To preserve consistency and trustworthiness, the Global Credit Cards dataset must make sure that crucial characteristics, such credit limits and payment histories, adhere to predetermined norms.

**Evaluation:**

* The SH Data Warehouse performs exceptionally well in terms of conformity, closely following defined data standards.
* The Global Credit Cards dataset needs to improve its adherence to data standards, especially where it matters most for evaluating credit risk.

**Conclusion**

In conclusion, there are advantages and disadvantages to both the SH Data Warehouse and the Global Credit Cards dataset in terms of standards and data quality. The data quality of the SH Data Warehouse is generally excellent, exhibiting high degrees of accuracy, completeness, consistency, and conformity. There is room for improvement, though, especially in terms of making sure that data is updated frequently to preserve timeliness.

Although helpful, the Global Credit Cards dataset still has to be significantly improved in terms of correctness, completeness, consistency, and conformity in order to be able to provide trustworthy credit evaluations. It's imperative to take care of these problems to prevent expensive mistakes, inefficiencies, and possible reputational harm to the company. The quality of these datasets must be continuously maintained and improved over time, requiring validation, adherence to best practices, and monitoring (Cai & Zhu, 2015).

**References**

Ambe, R. (2023) ‘Classification and quantification of timestamp data quality issues and its impact on data quality outcome’, *Data Intelligence*, pp. 1–39. doi:10.1162/dint\_a\_00238.

Cai, L. and Zhu, Y. (2015) ‘The challenges of data quality and data quality assessment in the Big Data Era’, *Data Science Journal*, 14(0), p. 2. doi:10.5334/dsj-2015-002.

Ehrlinger, L. and Wöß, W. (2022) ‘A survey of data quality measurement and Monitoring Tools’, *Frontiers in Big Data*, 5. doi:10.3389/fdata.2022.850611.

Ehsani-Moghaddam, B., Martin, K. and Queenan, J.A. (2019) ‘Data quality in Healthcare: A report of practical experience with the Canadian Primary Care Sentinel Surveillance Network Data’, *Health Information Management Journal*, 50(1–2), pp. 88–92. doi:10.1177/1833358319887743.

Even, A., Shankaranarayanan, G. and Berger, P.D. (2010) ‘Evaluating a model for cost-effective data quality management in a real-world CRM setting’, *Decision Support Systems*, 50(1), pp. 152–163. doi:10.1016/j.dss.2010.07.011.

Günther, L.C. et al. (2019) ‘Data quality assessment for improved decision-making: A methodology for small and medium-sized enterprises’, *Procedia Manufacturing*, 29, pp. 583–591. doi:10.1016/j.promfg.2019.02.114.

Hassenstein, M.J. and Vanella, P. (2022) ‘Data quality—concepts and problems’, *Encyclopedia*, 2(1), pp. 498–510. doi:10.3390/encyclopedia2010032.

Ramasamy, A. and Chowdhury, S. (2020) ‘Big Data Quality Dimensions: A systematic literature review’, *Journal of Information Systems and Technology Management* [Preprint]. doi:10.4301/s1807-1775202017003.

Zhu, H., Lee, Y.W. and Rosenthal, A.S. (2016) ‘Data standards challenges for interoperable and Quality Data’, *Journal of Data and Information Quality*, 7(1–2), pp. 1–3. doi:10.1145/2903723.

Elmasri, R., & Navathe, S. B. (2015). Fundamentals of Database Systems (7th ed.). Pearson.

Inmon, W. H. (2005). Building the Data Warehouse (4th ed.). John Wiley & Sons.

Kimball, R., & Ross, M. (2013). The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling (3rd ed.). John Wiley & Sons.

Oracle. (2023). Oracle Database Documentation. Retrieved from https://docs.oracle.com/en/database/

Ramakrishnan, R., & Gehrke, J. (2003). Database Management Systems (3rd ed.). McGraw-Hill.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2011). Database System Concepts (6th ed.). McGraw-Hill.

Potineni, P. (2021). *Oracle Database Data Warehousing Guide*, 19c. Oracle Corporation. Available at: <https://docs.oracle.com/en/database/oracle/oracle-database/19/dwhsg/>

Surampudi, S. (2017). *Data Mining User's Guide*, 12c Release 1 (12.1). Oracle Corporation. Available at: <https://docs.oracle.com/database/121/DMPRG/toc.htm>

**References & Bibliography**

Northumbria (2023) Academic Regulations for Taught Awards (ARTA) – 2022/23. Available at: <https://www.northumbria.ac.uk/about-us/university-services/student-library-and-academic-services/quality-and-teaching-excellence/assessment/assessment-regulations-and-policies/> (Accessed: 30 September 2023).

Pears, R. and  Shields, G. (2008) *Cite them right: the essential referencing guide*.   
Newcastle upon Tyne: Pear Tree Books. Available at: <https://www.citethemrightonline.com/> (Accessed: 16 October 2023).

Potineni, P. (2021) Oracle Database Data Warehousing Guide, 19c. Part Number E96243-05. Available at: <https://docs.oracle.com/en/database/oracle/oracle-database/19/dwhsg/> (Accessed: 25 October 2023).

Surampudi, S. (2017a) Data Mining User's Guide, 12c Release 1 (12.1). Part Number E53115-05. Available at: <https://docs.oracle.com/database/121/DMPRG/toc.htm> (Accessed: 16 October 20223).

Surampudi, S. (2017b) Oracle Data Mining Concepts, 12c Release 1 (12.1). Part Number E17692-19. Available at: <https://docs.oracle.com/database/121/DMCON/toc.htm> (Accessed: 16 October 2023).