Northumbria University
NEWCASTLE

**Department of Computer & Information Sciences**

# ASSESSMENT BRIEF

| | |
|---|---|
| **Module Title:** | Advanced Databases |
| **Module Code:** | KL7011 |
| **Academic Year / Semester:** | 2022-23 / Semester 1 |
| **Module Tutor / Email (all queries):** | Akhtar Ali 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:** | 25th November 2022 |
| **Mechanism for Handout:** | Module Blackboard Site & Live Session in Week 9 |
| **Deadline for Submission Attempt by Students:** | 19th January 2023 @ 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:** | 20th February 2023 |
| **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) | W21051498 / dwu57 / dmu115<br>W21055167/dwu139/dmu115 |
| Names of students in the Group | Adeoye Elijah<br>Temitope Olofe |
| Group No | 26 |

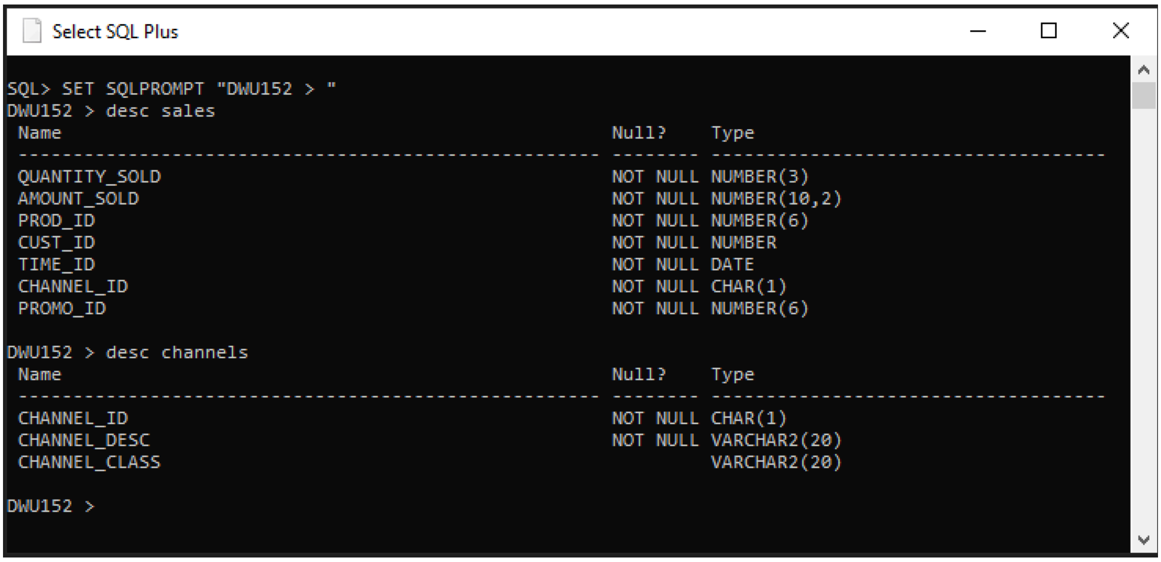
## Instructions on Assessment:

- 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 username (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.



```
SQL> SET SQLPROMPT "DWU152 > "
DWU152 > desc sales
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 QUANTITY_SOLD                                         NOT NULL NUMBER(3)
 AMOUNT_SOLD                                           NOT NULL NUMBER(10,2)
 PROD_ID                                               NOT NULL NUMBER(6)
 CUST_ID                                               NOT NULL NUMBER
 TIME_ID                                               NOT NULL DATE
 CHANNEL_ID                                            NOT NULL CHAR(1)
 PROMO_ID                                              NOT NULL NUMBER(6)

DWU152 > desc channels
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------------------------------
 CHANNEL_ID                                            NOT NULL CHAR(1)
 CHANNEL_DESC                                          NOT NULL VARCHAR2(20)
 CHANNEL_CLASS                                                  VARCHAR2(20)

DWU152 >
```

## 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).***

(A) 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, 2017) 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 discuss the differences in the performance of your queries with and without the proposed indexes. You need to cite relevant database literature to support your choice of indexes and how you dealt with the issue of selectivity in your queries.

(25 marks)

## Answer Part 1 (A)

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

```
DWU139> CREATE INDEX index_products
  2  ON products (prod_id, prod_name);

Index created.

Elapsed: 00:00:00.04
DWU139>
DWU139> CREATE INDEX index_sales
  2  ON sales (prod_id, quantity_sold, amount_sold);

Index created.

Elapsed: 00:00:01.89
DWU139>
DWU139> CREATE INDEX index_costs
  2  ON costs (prod_id, unit_cost, unit_price);

Index created.

Elapsed: 00:00:01.51
DWU139>
DWU139> CREATE INDEX index_customers
  2  ON customers (cust_id, cust_first_name);

Index created.

Elapsed: 00:00:00.08
DWU139>
DWU139> CREATE INDEX index_times
  2  ON times (time_id, day_name);

Index created.

Elapsed: 00:00:00.01
```

**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 (5 Marks):**

The index_sales, index_products and index_cost indexes were created as a performance tool because of the time it takes to produce an output. Indexes are fast partly because they do not have to carry all the data for each row in the table, just the data that we are looking for (Petrovic, 2018). The output for the 2 queries below shows the cost and time before the indexes were created.

**Provide 2 SQL queries you are going to run to compare the performance impact of your own 2 new indexes on DWU (6 marks).** ***Make sure the SQL code you provide is plain text.***

**Query 1:**

```
SELECT p.prod_id, p.prod_name, ct.unit_price,
s.quantity_sold, sum(s.amount_sold) sum_amount
FROM sales s,
     products p,
     costs ct
WHERE s.prod_id= p.prod_id
AND ct.prod_id = p.prod_id
GROUP BY p.prod_id, p.prod_name, ct.unit_price, s.quantity_sold;
```

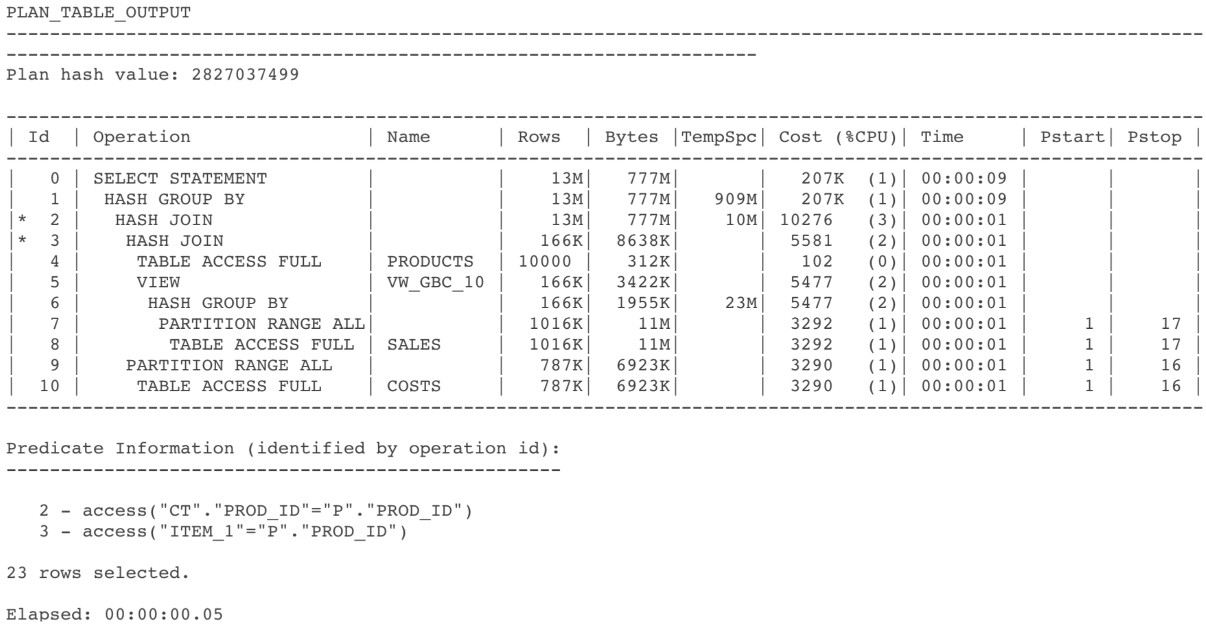
**Query 2:**

```
SELECT   t.time_id,
t.day_name,
count(cu.cust_first_name) number_customer,
sum(s.quantity_sold) sum_sold
FROM      sales s,
          customers cu,
          times t
WHERE     s.time_id = t.time_id
AND      cu.cust_id = s.cust_id
GROUP BY  t.time_id, t.day_name ;
```

**Provide Explain Plan 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.***

## Query 1: Before index was created

```
DWU139> set echo on
DWU139>
DWU139> EXPLAIN PLAN FOR
  2  SELECT p.prod_id, p.prod_name, ct.unit_price
  3  , s.quantity_sold, sum(s.amount_sold) sum_amount
  4  FROM sales s
  5  , products p
  6  , costs ct
  7  WHERE s.prod_id= p.prod_id
  8    AND ct.prod_id = p.prod_id
  9  GROUP BY p.prod_id, p.prod_name, ct.unit_price,
s.quantity_sold;

Explained.

Elapsed: 00:00:00.05
DWU139> select * from table(dbms_xplan.display());
```

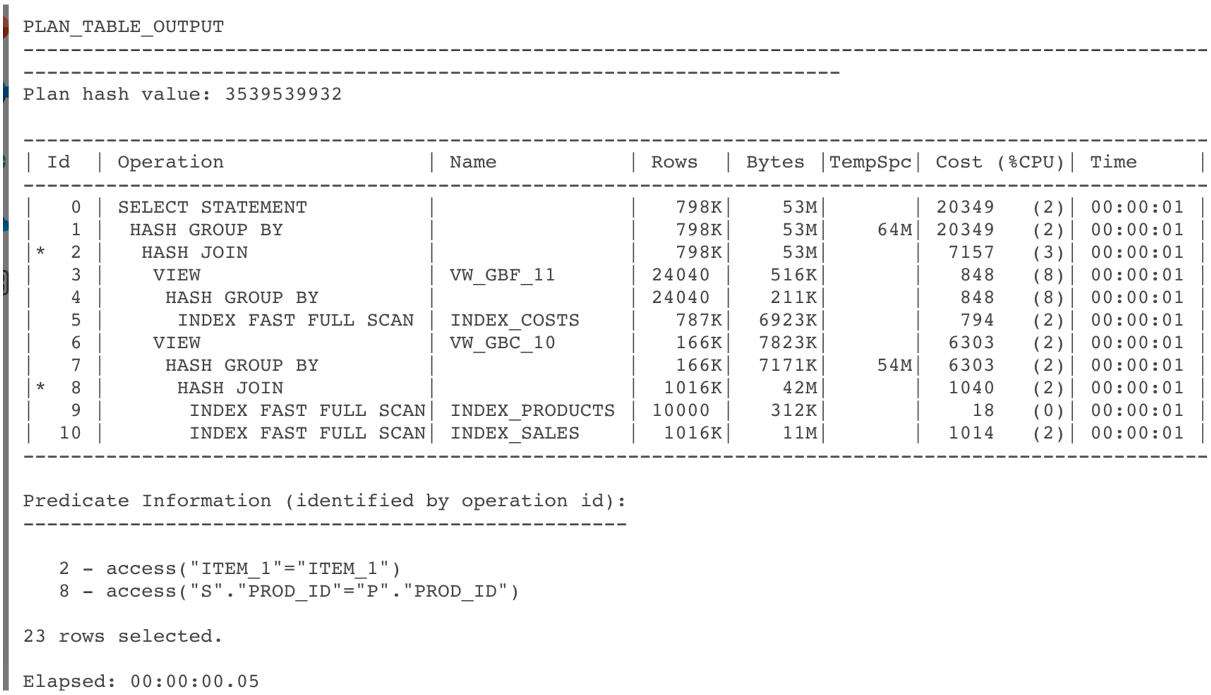


```
PLAN_TABLE_OUTPUT
----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------
Plan hash value: 2827037499

----------------------------------------------------------------------------------------------------------
| Id  | Operation                 | Name       | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     | Pstart| Pstop |
----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT          |            |   13M|   777M|       |   207K  (1)| 00:00:09 |       |       |
|   1 |  HASH GROUP BY            |            |   13M|   777M|   909M|   207K  (1)| 00:00:09 |       |       |
|*  2 |   HASH JOIN               |            |   13M|   777M|    10M| 10276   (3)| 00:00:01 |       |       |
|*  3 |    HASH JOIN              |            |  166K|  8638K|       |  5581   (2)| 00:00:01 |       |       |
|   4 |     TABLE ACCESS FULL     | PRODUCTS   | 10000 |   312K|       |   102   (0)| 00:00:01 |       |       |
|   5 |     VIEW                  | VW_GBC_10  |  166K|  3422K|       |  5477   (2)| 00:00:01 |       |       |
|   6 |      HASH GROUP BY        |            |  166K|  1955K|    23M|  5477   (2)| 00:00:01 |       |       |
|   7 |       PARTITION RANGE ALL|            | 1016K|    11M|       |  3292   (1)| 00:00:01 |     1 |    17 |
|   8 |        TABLE ACCESS FULL  | SALES      | 1016K|    11M|       |  3292   (1)| 00:00:01 |     1 |    17 |
|   9 |    PARTITION RANGE ALL    |            |  787K|  6923K|       |  3290   (1)| 00:00:01 |     1 |    16 |
|  10 |     TABLE ACCESS FULL     | COSTS      |  787K|  6923K|       |  3290   (1)| 00:00:01 |     1 |    16 |
----------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("CT"."PROD_ID"="P"."PROD_ID")
   3 - access("ITEM_1"="P"."PROD_ID")

23 rows selected.

Elapsed: 00:00:00.05
```

## Query 1: After index was created

```
DWU139> set echo on
DWU139>
DWU139> EXPLAIN PLAN FOR
  2  SELECT p.prod_id, p.prod_name, ct.unit_price
  3  , s.quantity_sold, sum(s.amount_sold) sum_amount
  4  FROM sales s
  5  , products p
  6  , costs ct
  7  WHERE s.prod_id= p.prod_id
  8    AND ct.prod_id = p.prod_id
  9  GROUP BY p.prod_id, p.prod_name, ct.unit_price,
s.quantity_sold;
```

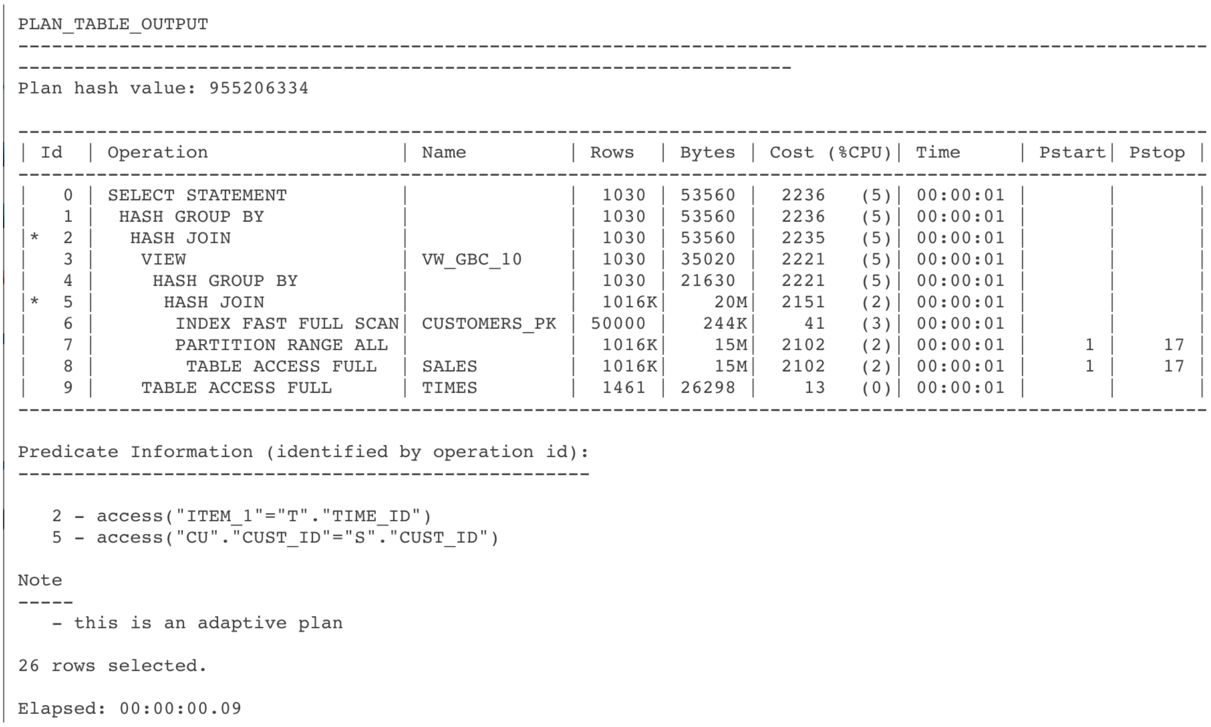
```
Explained.

Elapsed: 00:00:00.03
DWU139> select * from table(dbms_xplan.display());
```



```
PLAN_TABLE_OUTPUT
----------------------------------------------------------------------------------------------------
---------------------------------------------------------------------
Plan hash value: 3539539932

----------------------------------------------------------------------------------------------------
| Id  | Operation                   | Name           | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                |  798K|   53M|       | 20349   (2)| 00:00:01 |
|   1 |  HASH GROUP BY              |                |  798K|   53M|    64M| 20349   (2)| 00:00:01 |
|*  2 |   HASH JOIN                 |                |  798K|   53M|       |  7157   (3)| 00:00:01 |
|   3 |    VIEW                     | VW_GBF_11      | 24040 |  516K|       |   848   (8)| 00:00:01 |
|   4 |     HASH GROUP BY           |                | 24040 |  211K|       |   848   (8)| 00:00:01 |
|   5 |      INDEX FAST FULL SCAN   | INDEX_COSTS    |  787K| 6923K|       |   794   (2)| 00:00:01 |
|   6 |    VIEW                     | VW_GBC_10      |  166K| 7823K|       |  6303   (2)| 00:00:01 |
|   7 |     HASH GROUP BY           |                |  166K| 7171K|    54M|  6303   (2)| 00:00:01 |
|*  8 |      HASH JOIN              |                | 1016K|   42M|       |  1040   (2)| 00:00:01 |
|   9 |       INDEX FAST FULL SCAN| INDEX_PRODUCTS | 10000 |  312K|       |    18   (0)| 00:00:01 |
|  10 |       INDEX FAST FULL SCAN| INDEX_SALES    | 1016K|   11M|       |  1014   (2)| 00:00:01 |
----------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("ITEM_1"="ITEM_1")
   8 - access("S"."PROD_ID"="P"."PROD_ID")

23 rows selected.

Elapsed: 00:00:00.05
```

### Query 2: Before index was created

```
DWU139> set echo on
DWU139>
DWU139> EXPLAIN PLAN FOR
  2  SELECT   t.time_id
  3  ,         t.day_name
  4  ,         count(cu.cust_first_name) number_customer
  5  ,         sum(s.quantity_sold) sum_sold
  6  FROM      SH2.sales s
  7  ,         SH2.customers cu
  8  ,         SH2.times t
  9  WHERE     s.time_id = t.time_id
 10  AND     cu.cust_id = s.cust_id
 11  GROUP BY  t.time_id, t.day_name ;

Explained.

Elapsed: 00:00:00.07
DWU139>
DWU139>
DWU139>
DWU139> select * from table(dbms_xplan.display());
```



```
PLAN_TABLE_OUTPUT
----------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------
Plan hash value: 955206334

----------------------------------------------------------------------------------------------------------
| Id  | Operation                  | Name         | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT           |              |  1030 | 53560 |  2236   (5)| 00:00:01 |       |       |
|   1 |  HASH GROUP BY             |              |  1030 | 53560 |  2236   (5)| 00:00:01 |       |       |
|*  2 |   HASH JOIN                |              |  1030 | 53560 |  2235   (5)| 00:00:01 |       |       |
|   3 |    VIEW                    | VW_GBC_10    |  1030 | 35020 |  2221   (5)| 00:00:01 |       |       |
|   4 |     HASH GROUP BY          |              |  1030 | 21630 |  2221   (5)| 00:00:01 |       |       |
|*  5 |      HASH JOIN             |              |  1016K|    20M|  2151   (2)| 00:00:01 |       |       |
|   6 |       INDEX FAST FULL SCAN| CUSTOMERS_PK | 50000 |   244K|    41   (3)| 00:00:01 |       |       |
|   7 |       PARTITION RANGE ALL  |              |  1016K|    15M|  2102   (2)| 00:00:01 |     1 |    17 |
|   8 |        TABLE ACCESS FULL   | SALES        |  1016K|    15M|  2102   (2)| 00:00:01 |     1 |    17 |
|   9 |    TABLE ACCESS FULL       | TIMES        |  1461 | 26298 |    13   (0)| 00:00:01 |       |       |
----------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("ITEM_1"="T"."TIME_ID")
   5 - access("CU"."CUST_ID"="S"."CUST_ID")

Note
-----
   - this is an adaptive plan

26 rows selected.

Elapsed: 00:00:00.09
```

## Query 2: After the index was created

```
DWU139> set echo on
DWU139>
DWU139> EXPLAIN PLAN FOR
  2  SELECT   t.time_id
  3  ,         t.day_name
  4  ,         count(cu.cust_first_name) number_customer
  5  ,         sum(s.quantity_sold) sum_sold
  6  FROM      sales s
  7  ,         customers cu
  8  ,         times t
  9  WHERE     s.time_id = t.time_id
 10  AND     cu.cust_id = s.cust_id
 11  GROUP BY  t.time_id, t.day_name ;

Explained.

Elapsed: 00:00:00.02
DWU139>
DWU139> select * from table(dbms_xplan.display());
```

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------
---------------------------------------------------------------------
Plan hash value: 1943229767

------------------------------------------------------------------------------------------
| Id  | Operation                    | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                |  1030 | 53560 |  1319   (7)| 00:00:01 |
|   1 |  HASH GROUP BY               |                |  1030 | 53560 |  1319   (7)| 00:00:01 |
|*  2 |   HASH JOIN                  |                |  1030 | 53560 |  1318   (7)| 00:00:01 |
|   3 |    VIEW                      | VW_GBC_10      |  1030 | 35020 |  1315   (7)| 00:00:01 |
|   4 |     HASH GROUP BY            |                |  1030 | 21630 |  1315   (7)| 00:00:01 |
|*  5 |      HASH JOIN               |                |  1016K|    20M|  1245   (2)| 00:00:01 |
|   6 |       INDEX FAST FULL SCAN| CUSTOMERS_PK | 50000 |   244K|    43   (3)| 00:00:01 |
|   7 |       INDEX FAST FULL SCAN| INDEX_SALES  |  1016K|    15M|  1194   (2)| 00:00:01 |
|   8 |    INDEX FAST FULL SCAN   | INDEX_TIMES  |  1461 | 26298 |     3   (0)| 00:00:01 |
------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("ITEM_1"="T"."TIME_ID")
   5 - access("CU"."CUST_ID"="S"."CUST_ID")

21 rows selected.

Elapsed: 00:00:00.06
```

**Provide discussion of the cost-based comparison of the above 2 sets of queries and their explain plan cost figures/values (6 marks):**

For query 1, the explain plans are assigned a cost, which is an estimate of the time and resources required to execute that plan. Based on the explain plan above, after the application of the indexes which led to significant reduction in the cost on the CPU as well as the transaction time from 207k to 20349 on the select operation down to the last operation from 3290 to 1014. The optimizer picks the lowest cost plan and hands it off for execution.

For query 2, the explain plans are assigned a cost, which is an estimate of the time and resources required to execute that plan. Based on the explain plan above, after the application of the indexes which led to significant reduction in the cost on the CPU as well as the transaction time from 2236 to 1319 on the select operation down to the last operation from 13 to 3. The optimizer picks the lowest cost plan and hands it off for execution.

(B) 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 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, 2017) 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 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 cite relevant database literature to support your choice of MVs and queries.

(25 marks)

**Answer Part 1 (B)**

**Provide SQL code and output you used to create the 2 new MVs in your own DWU database (i.e., these MVs *must not* exist in SH2) (6 marks). *Make sure the SQL code you provide is plain text and the output is a screenshot.***

### Materialized view query 1

```
DWU139> CREATE MATERIALIZED VIEW famount_sold_mv
PCTFREE 5
BUILD IMMEDIATE
REFRESH COMPLETE
ENABLE QUERY REWRITE
AS
SELECT    p.prod_id
,         p.prod_name
,         ct.unit_price
,         s.quantity_sold
,         sum(s.amount_sold) AS Amount
FROM      sales s
,         products p
,         costs ct
WHERE     s.prod_id= p.prod_id
  AND     ct.prod_id = p.prod_id
GROUP BY  p.prod_id
,         p.prod_name
,         ct.unit_price
,         s.quantity_sold;

Materialized view created.
```

### Materialized view query 2

```
DWU139> CREATE MATERIALIZED VIEW day_name_amount_sold_mv
PCTFREE 5
BUILD IMMEDIATE
REFRESH COMPLETE
ENABLE QUERY REWRITE
AS
SELECT   t.time_id
,         t.day_name
,         count(cu.cust_first_name) number_customer
,         sum(s.quantity_sold) sum_sold
FROM      sales s
,         customers cu
,         times t
WHERE     s.time_id = t.time_id
  AND     s.cust_id = cu.cust_id
GROUP BY  t.time_id, t.day_name ;

Materialized view created.
```

**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 (5 Marks):**

A Materialized View is the result of a query on a table in your database that has been saved in memory or on disc so that you can easily access and query over its results in the future. It's a powerful tool that developers can use to improve query performance and simplify the development and maintenance of data products across a wide range of applications (Archer, 2022). The MVs were due to the frequently queried tables which are, sales, time, product, and cost. In order to speed up accessibility, the respective tables are saved in a materialized view for ease during future query operation. Using Materialized view saves cost.

**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 and the output is a screenshot.***

**QUERY 1**

```
SELECT p.prod_id, p.prod_name, ct.unit_price,
s.quantity_sold, sum(s.amount_sold) sum_amount
FROM sales s,
     products p,
     costs ct
WHERE s.prod_id= p.prod_id
AND ct.prod_id = p.prod_id
GROUP BY p.prod_id, p.prod_name, ct.unit_price, s.quantity_sold;
```

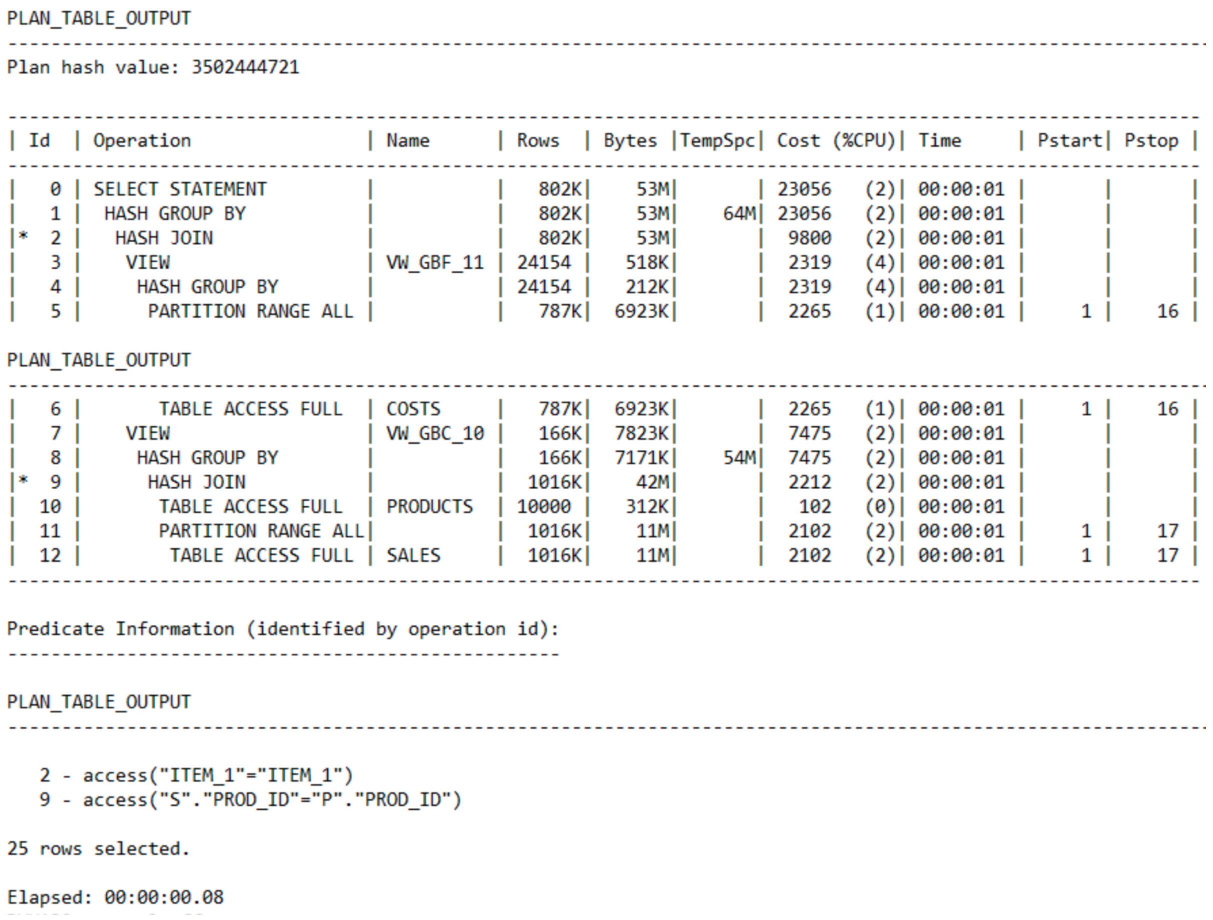
## QUERY 2

```
SELECT   t.time_id,
t.day_name,
count(cu.cust_first_name) number_customer,
sum(s.quantity_sold) sum_sold
FROM       sales s,
           customers cu,
           times t
WHERE      s.time_id = t.time_id
AND      cu.cust_id = s.cust_id
GROUP BY  t.time_id, t.day_name ;
```

**Provide <u>Explain Plan statements & outputs</u> for the above 3 SQL queries you have run to compare the performance impact of your 2 MVs on DWU and their version of the same queries on SH2 (4 marks):**

## QUERIES ON SH2 BEFORE MATERIALIZED VIEW

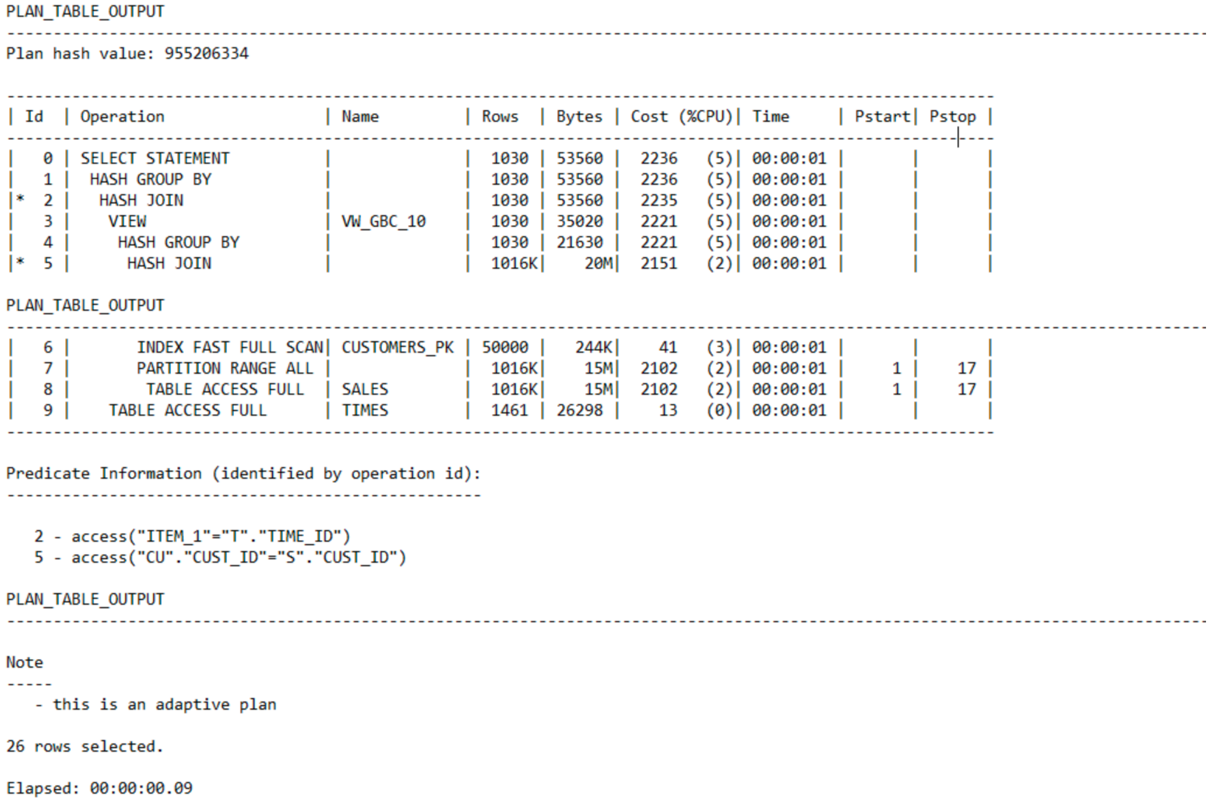
```
DWU139 > SET ECHO ON
DWU139 > SET TIMING ON
DWU139 >
DWU139 > EXPLAIN PLAN FOR
  2  SELECT p.prod_id, p.prod_name, ct.unit_price,
  3  s.quantity_sold, sum(s.amount_sold) sum_amount
  4  FROM SH2.sales s,
  5       SH2.products p,
  6       SH2.costs ct
  7  WHERE s.prod_id= p.prod_id
  8  AND ct.prod_id = p.prod_id
  9  GROUP BY p.prod_id, p.prod_name, ct.unit_price,
s.quantity_sold;

Explained.

Elapsed: 00:00:00.10
DWU139 > select * from table(dbms_xplan.display());
```



```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------
Plan hash value: 3502444721

-------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name     | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     | Pstart| Pstop |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |          |  802K|   53M|       | 23056   (2)| 00:00:01 |       |       |
|   1 |  HASH GROUP BY           |          |  802K|   53M|    64M| 23056   (2)| 00:00:01 |       |       |
|*  2 |   HASH JOIN              |          |  802K|   53M|       |  9800   (2)| 00:00:01 |       |       |
|   3 |    VIEW                  | VW_GBF_11 | 24154 |  518K|       |  2319   (4)| 00:00:01 |       |       |
|   4 |     HASH GROUP BY        |          | 24154 |  212K|       |  2319   (4)| 00:00:01 |       |       |
|   5 |      PARTITION RANGE ALL |          |  787K|  6923K|       |  2265   (1)| 00:00:01 |     1 |    16 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------
|   6 |       TABLE ACCESS FULL  | COSTS    |  787K|  6923K|       |  2265   (1)| 00:00:01 |     1 |    16 |
|   7 |    VIEW                  | VW_GBC_10 |  166K|  7823K|       |  7475   (2)| 00:00:01 |       |       |
|   8 |     HASH GROUP BY        |          |  166K|  7171K|    54M|  7475   (2)| 00:00:01 |       |       |
|*  9 |      HASH JOIN           |          | 1016K|   42M|       |  2212   (2)| 00:00:01 |       |       |
|  10 |       TABLE ACCESS FULL  | PRODUCTS | 10000 |  312K|       |   102   (0)| 00:00:01 |       |       |
|  11 |       PARTITION RANGE ALL|          | 1016K|   11M|       |  2102   (2)| 00:00:01 |     1 |    17 |
|  12 |        TABLE ACCESS FULL | SALES    | 1016K|   11M|       |  2102   (2)| 00:00:01 |     1 |    17 |
-------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------

   2 - access("ITEM_1"="ITEM_1")
   9 - access("S"."PROD_ID"="P"."PROD_ID")

25 rows selected.

Elapsed: 00:00:00.08
```

```
DWU139 > SET ECHO ON
DWU139 > SET TIMING ON
DWU139 >
DWU139 > EXPLAIN PLAN FOR
  2  SELECT   t.time_id,
  3  t.day_name,
  4  count(cu.cust_first_name) number_customer,
  5  sum(s.quantity_sold) sum_sold
  6  FROM       SH2.sales s,
  7             SH2.customers cu,
  8             SH2.times t
  9  WHERE      s.time_id = t.time_id
 10  AND       cu.cust_id = s.cust_id
 11  GROUP BY   t.time_id, t.day_name;

Explained.

Elapsed: 00:00:00.04
```

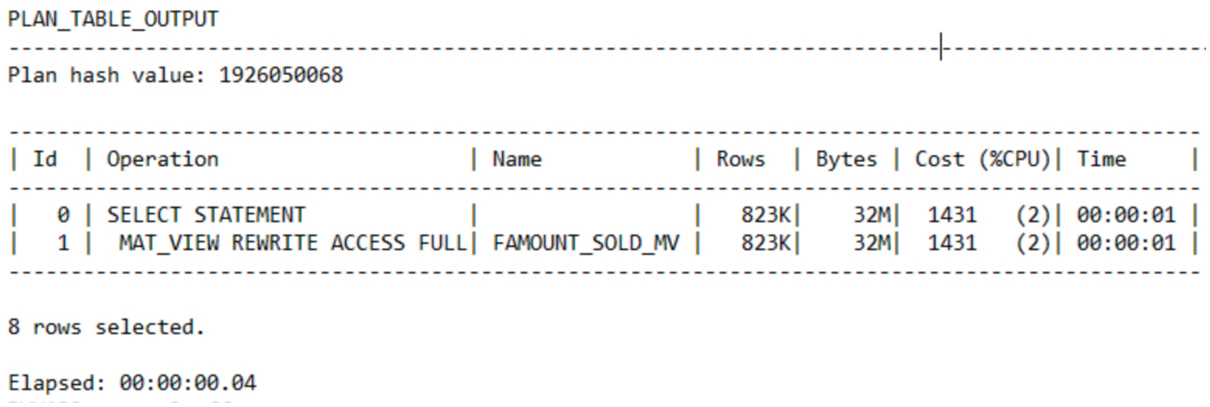
```
DWU139 >
DWU139 >
DWU139 > select * from table(dbms_xplan.display());
```



```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------
Plan hash value: 955206334

-----------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name          | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |               |  1030 | 53560 |  2236   (5)| 00:00:01 |       |       |
|   1 |  HASH GROUP BY               |               |  1030 | 53560 |  2236   (5)| 00:00:01 |       |       |
|*  2 |   HASH JOIN                  |               |  1030 | 53560 |  2235   (5)| 00:00:01 |       |       |
|   3 |    VIEW                      | VW_GBC_10     |  1030 | 35020 |  2221   (5)| 00:00:01 |       |       |
|   4 |     HASH GROUP BY            |               |  1030 | 21630 |  2221   (5)| 00:00:01 |       |       |
|*  5 |      HASH JOIN               |               |  1016K|    20M|  2151   (2)| 00:00:01 |       |       |

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------
|   6 |       INDEX FAST FULL SCAN| CUSTOMERS_PK | 50000 |   244K|    41   (3)| 00:00:01 |       |       |
|   7 |       PARTITION RANGE ALL |              |  1016K|    15M|  2102   (2)| 00:00:01 |     1 |    17 |
|   8 |        TABLE ACCESS FULL  | SALES        |  1016K|    15M|  2102   (2)| 00:00:01 |     1 |    17 |
|   9 |    TABLE ACCESS FULL      | TIMES        |  1461 | 26298 |    13   (0)| 00:00:01 |       |       |
-----------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("ITEM_1"="T"."TIME_ID")
   5 - access("CU"."CUST_ID"="S"."CUST_ID")

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------

Note
-----
   - this is an adaptive plan

26 rows selected.

Elapsed: 00:00:00.09
```

## QUERIES ON DWU AFTER MATERIALIZED VIEWS WERE CREATED

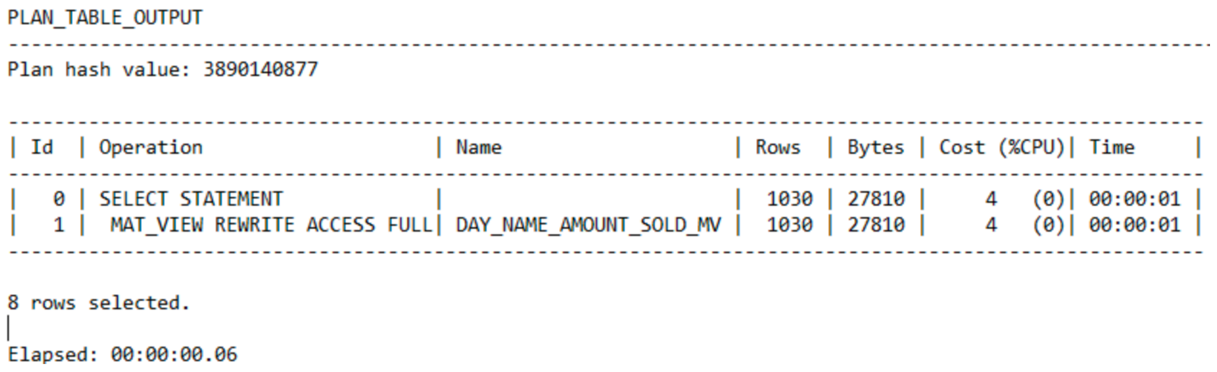
```
DWU139 > SET ECHO ON
DWU139 > SET TIMING ON
DWU139 >
DWU139 > EXPLAIN PLAN FOR
  2       SELECT    p.prod_id
  3       ,         p.prod_name
  4       ,         ct.unit_price
  5       ,         s.quantity_sold
  6       ,         sum(s.amount_sold) AS Amount
  7       FROM      sales s
  8       ,         products p
  9       ,         costs ct
 10      WHERE     s.prod_id= p.prod_id
 11        AND     ct.prod_id = p.prod_id
 12      GROUP BY  p.prod_id
 13     ,         p.prod_name
 14     ,         ct.unit_price
 15    ,         s.quantity_sold;

Explained.

Elapsed: 00:00:00.07
DWU139 >
DWU139 >
DWU139 > select * from table(dbms_xplan.display());
```



```
PLAN_TABLE_OUTPUT
---------------------------------------------------------------------------------------------
Plan hash value: 1926050068

--------------------------------------------------------------------------------------------
| Id  | Operation                     | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                |   823K|    32M|  1431   (2)| 00:00:01 |
|   1 |  MAT_VIEW REWRITE ACCESS FULL| FAMOUNT_SOLD_MV |   823K|    32M|  1431   (2)| 00:00:01 |
--------------------------------------------------------------------------------------------

8 rows selected.

Elapsed: 00:00:00.04
```

```
DWU139 > SET ECHO ON
DWU139 > SET TIMING ON
DWU139 >
DWU139 > EXPLAIN PLAN FOR
  2      SELECT   t.time_id
  3       ,         t.day_name
  4       ,         count(cu.cust_first_name) number_customer
  5       ,         sum(s.quantity_sold) sum_sold
  6      FROM       sales s
  7       ,         customers cu
  8       ,         times t
  9      WHERE      s.time_id = t.time_id
 10       AND      s.cust_id = cu.cust_id
 11    GROUP BY  t.time_id, t.day_name ;

Explained.

Elapsed: 00:00:00.03
```

```
DWU139 >
DWU139 >   select * from table(dbms_xplan.display());
```



```
PLAN_TABLE_OUTPUT
----------------------------------------------------------------------------------------------------------
Plan hash value: 3890140877

---------------------------------------------------------------------------------------------------------
| Id  | Operation                   | Name                     | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                          |  1030 | 27810 |     4   (0)| 00:00:01 |
|   1 |  MAT_VIEW REWRITE ACCESS FULL| DAY_NAME_AMOUNT_SOLD_MV |  1030 | 27810 |     4   (0)| 00:00:01 |
---------------------------------------------------------------------------------------------------------

8 rows selected.
|
Elapsed: 00:00:00.06
```

**Provide Discussion of the cost-based comparison of the above 3 sets of queries and their explain plan cost figures / values (6 marks):**

Because it stores pre-computed data, the materialised view is especially useful for improving query performance.

However, due to the view maintenance cost, all the views or queries are not candidates for materialisation. The important issue in data warehouse is the selection of views to materialise. For Query 1 execution, leads to a cost reduction of 1431 from 2265 with a byte reduction of 32M from 53M. Also, query 2 execution leads to a cost reduction of 4 from 2236 and a significant reduction in byte from 53560 to 27810.

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

This part is based on the UNITED FINANCE 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 UNITED FINANCE 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 UNITED FINANCE company's scenario. (5 marks)

Provide your answer here

The UNITED FINANCE problem belongs to a predictive data mining model since it involves the analysis of numeric data performance to forecast a future event, or trends. The 47 features as well as custid and defaultnm represents transactions where future predictions can be derived. By using supervised learning method, classification is preferred because of the nature of the target variable. It is a binary problem.

2. Prepare and setup your views and tables under your DMU account for accessing the shared `UnitedCreditCards` dataset, which also includes splitting the dataset for building, testing and applying the data mining models. (6 marks)

Provide whatever SQL code you have used for this part as plain TEXT and outputs as screenshots.

### Building the Data mining models

```
DMU115 > CREATE VIEW mining_data_credit_build_str_v(
  2  CUSTID,
  3  FEATURE1,
  4  FEATURE3,
  5  FEATURE4,
FEATURE5,
FEATURE9,
FEATURE11,
FEATURE12,
FEATURE13,
FEATURE14,
FEATURE16,
FEATURE17,
FEATURE18,
FEATURE19,
FEATURE20,
FEATURE21,
FEATURE22,
FEATURE23,
FEATURE24,
FEATURE28,
FEATURE33,
FEATURE34,
FEATURE35,
FEATURE36,
FEATURE37,
FEATURE38,
FEATURE39,
FEATURE40,
FEATURE41,
FEATURE42,
FEATURE43,
FEATURE44,
FEATURE45,
FEATURE46,
DEFAULTNM)
AS SELECT
u.CUSTID,
u.FEATURE1,
u.FEATURE3,
u.FEATURE4,
u.FEATURE5,
u.FEATURE9,
 44  u.FEATURE11,
 45  u.FEATURE12,
 46  u.FEATURE13,
 47  u.FEATURE14,
 48  u.FEATURE16,
 49  u.FEATURE17,
 50  u.FEATURE18,
 51  u.FEATURE19,
 52  u.FEATURE20,
 53  u.FEATURE21,
 54  u.FEATURE22,
 55  u.FEATURE23,
 56  u.FEATURE24,
 57  u.FEATURE28,
 58  u.FEATURE33,
 59  u.FEATURE34,
 60  u.FEATURE35,
 61  u.FEATURE36,
 62  u.FEATURE37,
 63  u.FEATURE38,
 64  u.FEATURE39,
 65  u.FEATURE40,
 66  u.FEATURE41,
 67  u.FEATURE42,
 68  u.FEATURE43,
 69  u.FEATURE44,
 70  u.FEATURE45,
 71  u.FEATURE46,
 72  u.defaultnm
 73  FROM
 74  unitedcreditcards u
 75  WHERE u.custid between 265001 and 290000;

View created.

DMU115 > CREATE VIEW mining_data_credit_build_v AS
  2  SELECT
  3   u.CUSTID,
u.FEATURE1,
u.FEATURE3,
u.FEATURE4,
u.FEATURE5,
u.FEATURE9,
u.FEATURE11,
u.FEATURE12,
u.FEATURE13,
u.FEATURE14,
u.FEATURE16,
u.FEATURE17,
u.FEATURE18,
u.FEATURE19,
u.FEATURE20,
u.FEATURE21,
u.FEATURE22,
u.FEATURE23,
u.FEATURE24,
u.FEATURE28,
u.FEATURE33,
u.FEATURE34,
u.FEATURE35,
u.FEATURE36,
u.FEATURE37,
u.FEATURE38,
u.FEATURE39,
u.FEATURE40,
u.FEATURE41,
u.FEATURE42,
u.FEATURE43,
u.FEATURE44,
u.FEATURE45,
u.FEATURE46,
u.defaultnm
FROM
unitedcreditcards u
WHERE u.custid between 265001 and 290000;
```

```
View created.

DMU115 > CREATE TABLE mining_build_credit AS
  2  SELECT
  3   u.CUSTID,
  4  u.FEATURE1,
  5  u.FEATURE3,
  6  u.FEATURE4,
  7  u.FEATURE5,
  8  u.FEATURE9,
  9  u.FEATURE11,
 10  u.FEATURE12,
 11  u.FEATURE13,
 12  u.FEATURE14,
 13  u.FEATURE16,
 14  u.FEATURE17,
 15  u.FEATURE18,
 16  u.FEATURE19,
 17  u.FEATURE20,
 18  u.FEATURE21,
 19  u.FEATURE22,
 20  u.FEATURE23,
 21  u.FEATURE24,
 22  u.FEATURE28,
 23  u.FEATURE33,
 24  u.FEATURE34,
 25  u.FEATURE35,
 26  u.FEATURE36,
 27  u.FEATURE37,
 28  u.FEATURE38,
 29  u.FEATURE39,
 30  u.FEATURE40,
 31  u.FEATURE41,
 32  u.FEATURE42,
 33  u.FEATURE43,
 34  u.FEATURE44,
 35  u.FEATURE45,
 36  u.FEATURE46,
 37  u.defaultnm
 38  FROM
 39  unitedcreditcards u
 40  WHERE u.custid between 265001 and 290000;

Table created.
```

### Testing the Data mining model

```
DMU115 > CREATE TABLE mining_test_credit AS
  2  SELECT
  3   u.CUSTID,
  4  u.FEATURE1,
  5  u.FEATURE3,
  6  u.FEATURE4,
  7  u.FEATURE5,
  8  u.FEATURE9,
  9  u.FEATURE11,
 10  u.FEATURE12,
 11  u.FEATURE13,
 12  u.FEATURE14,
 13  u.FEATURE16,
 14  u.FEATURE17,
 15  u.FEATURE18,
 16  u.FEATURE19,
 17  u.FEATURE20,
 18  u.FEATURE21,
 19  u.FEATURE22,
 20  u.FEATURE23,
 21  u.FEATURE24,
 22  u.FEATURE28,
 23  u.FEATURE33,
 24  u.FEATURE34,
 25  u.FEATURE35,
 26  u.FEATURE36,
 27  u.FEATURE37,
 28  u.FEATURE38,
 29  u.FEATURE39,
 30  u.FEATURE40,
 31  u.FEATURE41,
 32  u.FEATURE42,
 33  u.FEATURE43,
 34  u.FEATURE44,
 35  u.FEATURE45,
 36  u.FEATURE46,
 37  u.defaultnm
 38  FROM
 39  unitedcreditcards u
 40  WHERE u.custid between 290001 and 315000;

Table created.


DMU115 > CREATE VIEW mining_data_credit_test_v AS
  2  SELECT
  3   u.CUSTID,
  4  u.FEATURE1,
  5  u.FEATURE3,
  6  u.FEATURE4,
  7  u.FEATURE5,
  8  u.FEATURE9,
  9  u.FEATURE11,
 10  u.FEATURE12,
 11  u.FEATURE13,
 12  u.FEATURE14,
 13  u.FEATURE16,
 14  u.FEATURE17,
 15  u.FEATURE18,
 16  u.FEATURE19,
 17  u.FEATURE20,
 18  u.FEATURE21,
 19  u.FEATURE22,
 20  u.FEATURE23,
 21  u.FEATURE24,
 22  u.FEATURE28,
 23  u.FEATURE33,
 24  u.FEATURE34,
 25  u.FEATURE35,
 26  u.FEATURE36,
 27  u.FEATURE37,
 28  u.FEATURE38,
 29  u.FEATURE39,
 30  u.FEATURE40,
 31  u.FEATURE41,
 32  u.FEATURE42,
 33  u.FEATURE43,
 34  u.FEATURE44,
 35  u.FEATURE45,
 36  u.FEATURE46,
 37  u.defaultnm
 38  FROM
 39  unitedcreditcards u
 40  WHERE u.custid between 290001 and 315000;

View created.
```

### Applying the Data mining model

```
DMU115 > CREATE VIEW mining_data_credit_apply_str_v(
  2  CUSTID,
  3  FEATURE1,
  4  FEATURE3,
  5  FEATURE4,
  6  FEATURE5,
  7  FEATURE9,
  8  FEATURE11,
  9  FEATURE12,
 10  FEATURE13,
 11  FEATURE14,
 12  FEATURE16,
 13  FEATURE17,
 14  FEATURE18,
 15  FEATURE19,
 16  FEATURE20,
FEATURE21,
FEATURE22,
FEATURE23,
FEATURE24,
FEATURE28,
FEATURE33,
FEATURE34,
FEATURE35,
FEATURE36,
FEATURE37,
FEATURE38,
FEATURE39,
FEATURE40,
FEATURE41,
FEATURE42,
FEATURE43,
FEATURE44,
FEATURE45,
FEATURE46,
DEFAULTNM)
AS SELECT
u.CUSTID,
u.FEATURE1,
u.FEATURE3,
u.FEATURE4,
u.FEATURE5,
u.FEATURE9,
u.FEATURE11,
u.FEATURE12,
u.FEATURE13,
u.FEATURE14,
u.FEATURE16,
u.FEATURE17,
u.FEATURE18,
u.FEATURE19,
u.FEATURE20,
u.FEATURE21,
u.FEATURE22,
 55  u.FEATURE23,
 56  u.FEATURE24,
 57  u.FEATURE28,
 58  u.FEATURE33,
 59  u.FEATURE34,
 60  u.FEATURE35,
 61  u.FEATURE36,
 62  u.FEATURE37,
 63  u.FEATURE38,
 64  u.FEATURE39,
 65  u.FEATURE40,
 66  u.FEATURE41,
 67  u.FEATURE42,
 68  u.FEATURE43,
 69  u.FEATURE44,
 70  u.FEATURE45,
 71  u.FEATURE46,
 72  u.defaultnm
 73  FROM
 74  unitedcreditcards u
 75  WHERE u.custid between 240000 and 265000;

View created.


DMU115 > CREATE or replace VIEW mining_data_credit_apply_v AS
  2  SELECT
  3  u.CUSTID,
  4  u.FEATURE1,
  5  u.FEATURE3,
  6  u.FEATURE4,
  7  u.FEATURE5,
  8  u.FEATURE9,
  9  u.FEATURE11,
 10  u.FEATURE12,
 11  u.FEATURE13,
 12  u.FEATURE14,
 13  u.FEATURE16,
 14  u.FEATURE17,
 15  u.FEATURE18,
 16  u.FEATURE19,
 17  u.FEATURE20,
 18  u.FEATURE21,
 19  u.FEATURE22,
 20  u.FEATURE23,
 21  u.FEATURE24,
 22  u.FEATURE28,
 23  u.FEATURE33,
 24  u.FEATURE34,
 25  u.FEATURE35,
 26  u.FEATURE36,
 27  u.FEATURE37,
 28  u.FEATURE38,
 29  u.FEATURE39,
 30  u.FEATURE40,
 31  u.FEATURE41,
 32  u.FEATURE42,
 33  u.FEATURE43,
 34  u.FEATURE44,
 35  u.FEATURE45,
 36  u.FEATURE46,
 37  u.defaultnm
 38  FROM
 39  unitedcreditcards u
 40  WHERE u.custid between 240000 and 265000;

View created.



DMU115 > CREATE TABLE mining_apply_credit AS
  2  SELECT
  3   u.CUSTID,
  4  u.FEATURE1,
  5  u.FEATURE3,
  6  u.FEATURE4,
  7  u.FEATURE5,
  8  u.FEATURE9,
  9  u.FEATURE11,
 10  u.FEATURE12,
 11  u.FEATURE13,
 12  u.FEATURE14,
 13  u.FEATURE16,
 14  u.FEATURE17,
 15  u.FEATURE18,
 16  u.FEATURE19,
 17  u.FEATURE20,
 18  u.FEATURE21,
 19  u.FEATURE22,
 20  u.FEATURE23,
 21  u.FEATURE24,
 22  u.FEATURE28,
 23  u.FEATURE33,
 24  u.FEATURE34,
 25  u.FEATURE35,
 26  u.FEATURE36,
 27  u.FEATURE37,
 28  u.FEATURE38,
 29  u.FEATURE39,
 30  u.FEATURE40,
 31  u.FEATURE41,
 32  u.FEATURE42,
 33  u.FEATURE43,
 34  u.FEATURE44,
 35  u.FEATURE45,
 36  u.FEATURE46,
 37  u.defaultnm
 38  FROM
 39  unitedcreditcards u
 40  WHERE u.custid between 240000 and 265000;

Table created.
```

### For Support Vector Machine Model

```
DMU115 > CREATE TABLE svm_model_settings (
  2    setting_name  VARCHAR2(30),
  3    setting_value VARCHAR2(30));

Table created.
```

### For Naïve Bayes model

```
DMU115 >   CREATE TABLE nbyes_model_settings (
  2        setting_name  VARCHAR2(30),
  3        setting_value VARCHAR2(30));

Table created.
```

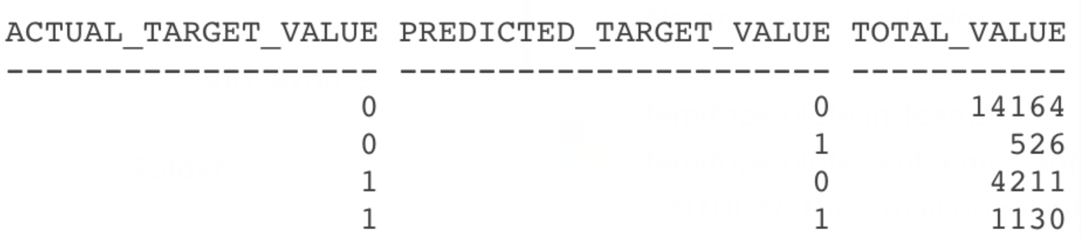
3. Using the PL/SQL Data Mining API, develop at least TWO models using suitable algorithms for performing your chosen data mining task on the `UnitedCreditCards` dataset.

(12 marks)

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

### For Support Vector Machine Model

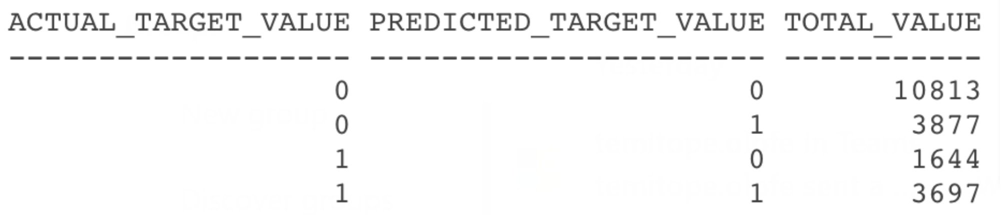
```
DMU115 > BEGIN
  2    INSERT INTO svm_model_settings (setting_name, setting_value)
VALUES
  3       (dbms_data_mining.algo_name,
dbms_data_mining.algo_support_vector_machines);
  4    INSERT INTO svm_model_settings (setting_name, setting_value)
VALUES
  5     (dbms_data_mining.prep_auto,dbms_data_mining.prep_auto_on);
  6    COMMIT;
  7  END;
  8  /

PL/SQL procedure successfully completed.

DMU115 >
DMU115 > BEGIN
  2    DBMS_DATA_MINING.CREATE_MODEL(
  3      model_name          => 'svm_model',
  4      mining_function     => dbms_data_mining.classification,
  5      data_table_name     => 'mining_data_credit_build_v',
  6      case_id_column_name => 'custid',
  7      target_column_name  => 'defaultnm',
  8      settings_table_name => 'svm_model_settings');
  9  END;
 10  /

PL/SQL procedure successfully completed.

DMU115 > SELECT defaultnm AS actual_target_value,
  2         PREDICTION(svm_model USING *) AS predicted_target_value,
  3         COUNT(*) AS total_value
  4  FROM mining_data_credit_test_v
  5  GROUP BY defaultnm, PREDICTION(svm_model USING *)
  6  ORDER BY 1, 2;
```



```
ACTUAL_TARGET_VALUE PREDICTED_TARGET_VALUE TOTAL_VALUE
------------------- ---------------------- -----------
                  0                      0       14164
                  0                      1         526
                  1                      0        4211
                  1                      1        1130
```

### For Naïve Bayes model

```
DMU115 > BEGIN
  2       INSERT INTO  nbyes_model_settings VALUES
  3           (dbms_data_mining.algo_name,
  4            dbms_data_mining.ALGO_NAIVE_BAYES);
  5         INSERT INTO  nbyes_model_settings  VALUES
  6          (dbms_data_mining.prep_auto,
  7           dbms_data_mining.prep_auto_on);
  8         COMMIT;
  9       END;
 10     /

PL/SQL procedure successfully completed.

DMU115 > BEGIN
  2         DBMS_DATA_MINING.CREATE_MODEL(
  3           model_name          => 'nbyes_model',
  4           mining_function     => dbms_data_mining.classification,
  5           data_table_name     => 'mining_data_credit_build_v',
  6           case_id_column_name => 'custid',
  7           target_column_name  => 'defaultnm',
  8           settings_table_name => 'nbyes_model_settings');
  9       END;
 10     /

PL/SQL procedure successfully completed.

DMU115 > SELECT defaultnm AS actual_target_value,
  2              PREDICTION(nbyes_model USING *) AS
predicted_target_value,
  3              COUNT(*) AS total_value
  4       FROM mining_data_credit_test_v
  5       GROUP BY defaultnm, PREDICTION(nbyes_model USING *)
  6      ORDER BY 1, 2;
```



```
ACTUAL_TARGET_VALUE PREDICTED_TARGET_VALUE TOTAL_VALUE
------------------- ---------------------- -----------
                  0                      0       10813
                  0                      1        3877
                  1                      0        1644
                  1                      1        3697
```

4. Evaluate capabilities of the models you have developed for this task.

(6 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.

**For Support Vector Machine Model**

```
DMU115 > COLUMN ACCURACY FORMAT 99.99
DMU115 > SELECT (SUM(correct)/COUNT(*))*100  AS accuracy
  2      FROM  (SELECT DECODE(defaultnm,
  3            PREDICTION(svm_model USING *), 1, 0) AS correct
  4           FROM mining_data_credit_test_v);

ACCURACY
--------
   76.35
```

**For Naïve Bayes model**

```
DMU115 > SELECT (SUM(correct)/COUNT(*))*100  AS accuracy
  2     FROM  (SELECT DECODE(defaultnm,
  3                          PREDICTION(nbyes_model USING *), 1, 0)
AS correct
  4                     FROM mining_data_credit_test_v);

ACCURACY
--------
   72.44
```

5. Present and discuss your findings and make recommendations to the Managing Director of UNITED FINANCE company.

(6 marks)

Provide your answer here

The findings from the united finance company dataset showed that some of the features have similar meaning and we felt merging them together would have produced better result and accuracy. A situation where we have features referring to same similar credit type or line needs improvement. From the analysis, SVM performed better than naïve bayes with an accuracy of 76.35% for the 20,031 rows compared to 72.44%. Recall that SVM is very good and efficient for classification problem like binary classification while Naïve Bayes is good for text classification.

It can be concluded that the dataset would require additional feature vectors which were not included such year of loan etc, Attempting to generalise a subspace of the actual input space in which the other dimensions were unknown, and as a result, SVM performed better than Naïve Bayes If similar analysis were conducted in the future to generate the dataset used in this report, more feature vectors must be calculated so that the model can form a better understanding of the problem at hand.

**Part 3 (15 marks)**

***Critically evaluate the SH data warehouse and the UNITED FINANCE company's*** `UnitedCreditCards` ***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]

Before we evaluate these data sets in relation to the theory and best practices of data quality and standards, it important we briefly understand the basis of data quality and standards. Data is said to be of high-quality if it is fit for its purposes in operations, decision making, and planning (Ref). This means that they are fit for use when they free of imperfection and possess the characteristics required to complete the operations, make the decision, and complete the plan (Juran and Godfrey, 1998). Data standard on the other hand refers to a documented agreement or technical specification that states how data should be stored, updated, and exchanged consistently.

To ensure that data is reliable, it is paramount to understand the concept of data quality dimensions. Data quality dimensions are features which we use to measure data quality and will assist in assessing if data is good to use or may require improvements. Our evaluation of the SH data warehouse and the unitedcreditcards dataset will be based on the dimensions of data quality which are accuracy, completeness, consistency, timeliness, validity, uniqueness, and conformance.

The two datasets have required information for operations, decision making and planning and this meets the completeness dimension of data quality. We can make decision as well as planning based on the two datasets and can be said to meet the theory of best practices. Moreso, we do not have a missing value in these data sets which would have generated severe consequences in terms of making a valid decision or planning and thereby affecting the data quality. Data is said to be complete when all information required for a particular use is present and available for users. However, we should mistake completeness with accuracy as a complete data set might still have incorrect values.

When we talk about uniqueness dimension of data quality, we are simply referring to duplication in data. By duplicate, we are referring to storing or having same information more than once in a data. This is a serious risk because having duplicates in data sets will affect the analysis and thereby producing reports that cannot be said to be valid. A closer look at both datasets do not indicate any duplicate of same information which would have affected the quality of these datasets in terms of making accurate decision, planning and operations.

The availability of information is an essential part of data. Here, we are talking about timeliness. We are interested in knowing if required information is available to enable us carry out operation, make decision and complete a plan or not. Again, we can say these data sets have required information that we may need for analysis. For instance, the SH data set cover transactions in such a way that if we want to make analysis based on a given day of the week, we can get it. This timeliness dimension ensures that information is available at the appropriate time and thereby guaranteed data quality. Timeliness is very important because it adds value to record that is particularly time sensitive (Open risk manual, 2018).

Accuracy dimension of data quality refers to how well data reflects reality. For instance, this may mean having correct day of the week is such a way that it reflects reality. A high data accuracy produces a good analytical result which we can trust and in turn generates accurate decision. To a very large extent, the data sets can be said to be accurate because they both show some resemblance of reality in them which would enhance good analytical result and consequently leads to confident decision making. Kindly note that a review is needed for real world information of data that can change overtime. This is because for data that are personal, a change in personal circumstances might affect the accuracy of the data set and consequently renders it useless.

In terms of validity dimension of data quality, we are interested in knowing if our data are in specific format, type, and range. For instance, to capture weekly information, we must have days from Monday to Sunday in our data for us to consider it to be in the required format or an email must have @ included in it otherwise we cannot conclude that it is valid. Generally speaking, when we have a valid data, it means we can synchronize with other sources. Again, the information on the two data sets was in required format and type. This again shows some level of quality and standard in the data sets.

Furthermore, consistency is a very important dimension in data quality. Consistency is guaranteed when data values conform with other values within a record. This means that there is not conflict in values across different data sets. Consistency in data promotes the ability to synchronize data from multiple sources. Take for instance customer id or date of birth in two different data sets for same person should be same to guarantee data utility. These two data sets show some level of consistent and thereby can be said to follow theory of best practices (Gov.uk, 2021).

Finally, when we talk about conformance, we are looking at a situation whereby our data is in line with internal or external standards, ethics, or reference data. We want to sure if the data actually exist. From our finding, this information actually exists and not a fabrication. Hence, we can agree that it meets the theory of best practices in terms of data quality and standard.

## References & Bibliography

Northumbria (2022) Academic Regulations for Taught Awards 2022/23. Available at: https://www.northumbria.ac.uk/about-us/university-services/student-library-and-academic-services/quality-and-teaching-excellence/assessment/guidance-for-students/ (Accessed: 16 October 2022).

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 2022).

Potineni, P. (2017) Oracle Database Data Warehousing Guide, 12c Release 1 (12.1). Part Number E41670-11. Available at: https://docs.oracle.com/database/121/DWHSG/ (Accessed: 16 October 2022).

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 2022).

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 2022).

Archer, C. (2022) What are Materialized Views (and why do they matter for realtime)?. Available at: https://www.tinybird.co/blog-posts/what-are-materialized-views-and-why-do-they-matter-for-realtime?utm_source=Google+Paid&utm_medium=cpc&utm_campaign=Search+DSA+Blog&utm_content=143603275573&utm_term=&gclid=Cj0KCQiAq5meBhCyARIsAJrtdr68xhwusONyvIVPrukvzQSF0nDX9PgDm9XcNz9G8Um4x4Psgbx2TrgaAuvUEALw_wcB (Accessed: 16 January 2023)

Gov.uk (2021) Meet the data quality dimensions. Available at: https://www.gov.uk/government/news/meet-the-data-quality-dimensions (Accessed: 18 January 2023)

Juran, J. and Godfrey, B. (1998) *Juran’s quality handbook*, 5th (ed.). New York, USA: McGraw-Hill Professional Publishing

Petrovic, B. (2018) SQL index overview and strategy. Available at: https://www.sqlshack.com/sql-index-overview-and-strategy/ (Accessed: 11 January 2023)

Open risk manual (2018) Data quality standards. Available at: https://www.openriskmanual.org/wiki/Data_Quality_Standards (Accessed: 13 January 2023)