# Data Warehouse Optimization & Predictive Analytics for Credit Risk and Sales Performance

## QUESTIONS AND ANSWER BASED

Name: Srinithi Anand Prem Anand
Email: srinuanand.p@gmail.com

***Part 1: Data Warehousing Tasks***

(A) Study the index definitions in `sh_idx.sql`. These indexes have already been created in SHV2. 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 SHV2 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 *two* aggregate functions. 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.

**<u>Answer Part 1 (A):</u>**

**Query 1:** Analyse Sales by Customer City and Product.

**SQL QUERY:**

```
CREATE INDEX IDX_SALES_CUST_CITY_PROD
ON SALES (CUST_ID, PROD_ID);
```

**OUPUT:**

```
DWU543> CREATE INDEX IDX_SALES_CUST_CITY_PROD
  2  ON SALES (CUST_ID, PROD_ID);

Index created.

DWU543>
```

**EXPLANATION:**

The query joins SALES, CUSTOMERS, and PRODUCTS. Current indexes partially cover the columns involved:

- CUST_ID in SALES and CUSTOMERS.
- PROD_ID in SALES and PRODUCTS.

**Query 2:** Analyse Promotional Impact by Channel

**SQL QUERY:**

```
CREATE INDEX IDX_SALES_PROMO_CHANNEL
ON SALES (PROMO_ID, CHANNEL_ID);
```

## OUTPUT:

```
DWU543> CREATE INDEX IDX_SALES_PROMO_CHANNEL
  2  ON SALES (PROMO_ID, CHANNEL_ID);

Index created.

DWU543>
```

**Query** 1: Sales by Customer City and Product

**QUERY**:

```
SELECT
    c.cust_city,
    p.prod_category,
    SUM(s.amount_sold) AS total_sales_amount
FROM
    sales s
JOIN customers c ON s.cust_id = c.cust_id
JOIN products p ON s.prod_id = p.prod_id
GROUP BY
    c.cust_city, p.prod_category;
```

## OUTPUT:

```
sqlplus

DWU543> SELECT
  2       c.cust_city,
  3       p.prod_category,
  4       SUM(s.amount_sold) AS total_sales_amount
  5   FROM
  6       sales s
  7   JOIN customers c ON s.cust_id = c.cust_id
  8   JOIN products p ON s.prod_id = p.prod_id
  9   GROUP BY
 10       c.cust_city, p.prod_category;

CUST_CITY                      PROD_CATEGORY                                      TOTAL_SALES_AMOUNT
------------------------------ -------------------------------------------------- ------------------
Warstein                       Men                                                           2343958
Murnau                         Women                                                       4059764.4
Asten                          Boys                                                         431466.8
Oran                           Men                                                         742866.75
Soest                          Boys                                                         222222.5
Frederikshavn                  Girls                                                           80540
Frederikshavn                  Women                                                        295228.6
Cranford                       Men                                                          410652.6
Didcot                         Women                                                         2815147
Didcot                         Girls                                                        485374.4
Halifax                        Boys                                                           389113
San Mateo                      Men                                                          894667.6
Pala                           Women                                                       367088.95
Blagnac                        Girls                                                        201213.9
Blagnac                        Men                                                         511521.55
Killarney                      Girls                                                        632610.7
Killarney                      Men                                                         2316940.1
Malakoff                       Girls                                                         61057.4
Lindau                         Women                                                       2387280.2
Hoofddorp                      Men                                                          415940.9
Delft                          Men                                                          746101.1
Schimmert                      Boys                                                           390982
Stockport                      Women                                                       492955.95
Stockport                      Girls                                                        135782.9
Birmingham                     Men                                                        1192486.25
Rotterdam                      Boys                                                         186250.9
Blaubeuren                     Girls                                                       225834.65
Cap-d'Agde, Le                 Boys                                                         119231.9
Helmond                        Women                                                       2511039.3
Fords Prairie                  Girls                                                         32689.2
Fords Prairie                  Boys                                                          16566.2
Groningen                      Men                                                         1320420.6
Groningen                      Boys                                                        379808.95
Waddinxveen                    Women                                                      1967051.25
Aix-les-Bains                  Men                                                          254912.4
```

```
sqlplus
Buckley                        Women                                                         819
White Plains                   Girls                                                        2648
Stamford                       Boys                                                        62871
North Druid Hills              Boys                                                     171867.4
North Druid Hills              Girls                                                      143713
Forestville                    Women                                                      186088
Cayuga                         Boys                                                      91727.8
Potsdam                        Women                                                   403434.65
Potsdam                        Men                                                      365949.1
Revel                          Women                                                   318131.25
Warsaw                         Girls                                                      6651.2
Holyrood                       Girls                                                    21738.75
Redbridge                      Boys                                                      91240.8
Cape Town                      Women                                                  1024818.75
Cape Town                      Men                                                     572917.95
Tokyo                          Women                                                   264003.35
Orangeville                    Men                                                      272716.8
Delhi                          Girls                                                       41638
Vilafranca del Penedes         Boys                                                       127141
Cap d'Antibes, Le              Men                                                      268567.6
Lakeside                       Girls                                                     29261.8
Bryant                         Girls                                                    89083.05
Bryant                         Women                                                    321461.3
Paris                          Men                                                         10631
Ulm                            Boys                                                    164805.45
Ulm                            Women                                                    934135.7
Sainte-Croix-du-Mont           Girls                                                     15321.1
Elba                           Men                                                      226364.1
Elba                           Women                                                   363314.85
Bielefeld                      Girls                                                     42476.6
Richmond-upon-Thames           Girls                                                        2897
Groesbeek                      Men                                                      224711.5
City of London                 Girls                                                     45940.3
Bremen                         Women                                                    174529.9
Bremen                         Boys                                                      42010.2
Dillsboro                      Boys                                                      60886.7
Whalepass                      Boys                                                     43727.25
North Hills                    Men                                                     240103.45
Bradford, IL                   Girls                                                       88496
Rock Creek                     Men                                                      220414.4
Bridgman                       Men                                                        219587

CUST_CITY                      PROD_CATEGORY                                  TOTAL_SALES_AMOUNT
------------------------------ -------------------------------------------------- ------------------
Bamberg                        Men                                                      235567.6

1844 rows selected.

DWU543>
```

## **Query** 2: Promotional Impact by Channel

## SQL QUERY:

```
SELECT pr.promo_category, ch.channel_desc, MAX(s.amount_sold) AS max_sale
FROM sales s
join promotions pr ON s.promo_id = pr.promo_id
join channels ch ON s.channel_id = ch.channel_id
GROUP BY
pr.promo_category, ch.channel_desc;
```

```
DWU543> SELECT pr.promo_category, ch.channel_desc, MAX(s.amount_sold) AS max_sale
  2  FROM sales s
  3  join promotions pr ON s.promo_id = pr.promo_id
  4  join channels ch ON s.channel_id = ch.channel_id
  5  GROUP BY
  6  pr.promo_category, ch.channel_desc;

PROMO_CATEGORY                 CHANNEL_DESC           MAX_SALE
------------------------------ -------------------- ----------
NO PROMOTION                   Catalog                   10710
flyer                          Partners                 8797.5
magazine                       Catalog                  5614.4
magazine                       Direct Sales               9996
magazine                       Tele Sales                 6480
NO PROMOTION                   Tele Sales                 7735
newspaper                      Catalog                    9954
TV                             Partners                   6930
newspaper                      Partners                  11376
post                           Direct Sales               9954
radio                          Tele Sales                 3796

PROMO_CATEGORY                 CHANNEL_DESC           MAX_SALE
------------------------------ -------------------- ----------
flyer                          Tele Sales                 4599
internet                       Catalog                  6898.5
post                           Partners                   9855
TV                             Catalog                  5584.5
internet                       Internet                 9697.6
post                           Internet                   7315
radio                          Partners                   9900
magazine                       Partners                 6956.4
NO PROMOTION                   Partners                  11060
newspaper                      Direct Sales              14931
flyer                          Direct Sales              10293

PROMO_CATEGORY                 CHANNEL_DESC           MAX_SALE
------------------------------ -------------------- ----------
radio                          Direct Sales             6754.5
post                           Tele Sales                 3555
radio                          Catalog                  4892.8
radio                          Internet                   6804
TV                             Internet                  10948
NO PROMOTION                   Internet                  14615
internet                       Partners                  10710
TV                             Direct Sales              14994
NO PROMOTION                   Direct Sales              14965
newspaper                      Tele Sales                 4977
internet                       Tele Sales               4819.5
```

```
magazine                       Tele Sales                 6480
NO PROMOTION                   Tele Sales                 7735
newspaper                      Catalog                    9954
TV                             Partners                   6930
newspaper                      Partners                  11376
post                           Direct Sales               9954
radio                          Tele Sales                 3796

PROMO_CATEGORY                 CHANNEL_DESC           MAX_SALE
------------------------------ -------------------- ----------
flyer                          Tele Sales                 4599
internet                       Catalog                  6898.5
post                           Partners                   9855
TV                             Catalog                  5584.5
internet                       Internet                 9697.6
post                           Internet                   7315
radio                          Partners                   9900
magazine                       Partners                 6956.4
NO PROMOTION                   Partners                  11060
newspaper                      Direct Sales              14931
flyer                          Direct Sales              10293

PROMO_CATEGORY                 CHANNEL_DESC           MAX_SALE
------------------------------ -------------------- ----------
radio                          Direct Sales             6754.5
post                           Tele Sales                 3555
radio                          Catalog                  4892.8
radio                          Internet                   6804
TV                             Internet                  10948
NO PROMOTION                   Internet                  14615
internet                       Partners                  10710
TV                             Direct Sales              14994
NO PROMOTION                   Direct Sales              14965
newspaper                      Tele Sales                 4977
internet                       Tele Sales               4819.5

PROMO_CATEGORY                 CHANNEL_DESC           MAX_SALE
------------------------------ -------------------- ----------
flyer                          Catalog                    9996
magazine                       Internet                 8869.5
newspaper                      Internet                  10948
internet                       Direct Sales              12087
TV                             Tele Sales               4972.5
post                           Catalog                  6043.5
flyer                          Internet                  14931

40 rows selected.

DWU543>
```

## EXPLAIN PLAN: Query 1

## SQL QUERY:

```
EXPLAIN PLAN FOR
SELECT
    c.cust_city,
    p.prod_category,
    SUM(s.amount_sold) AS total_sales_amount
FROM
    sales s
JOIN customers c ON s.cust_id = c.cust_id
JOIN products p ON s.prod_id = p.prod_id
GROUP BY
    c.cust_city, p.prod_category;
```

## OUTPUT:

```
DWU543> EXPLAIN PLAN FOR
SELECT
    c.cust_city,
    p.prod_category,
    SUM(s.amount_sold) AS total_sales_amount
FROM
    sales s
JOIN customers c ON s.cust_id = c.cust_id
JOIN products p ON s.prod_id = p.prod_id
GROUP BY
    c.cust_city, p.prod_category;

Explained.

DWU543>
```

## SQL QUERY:

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

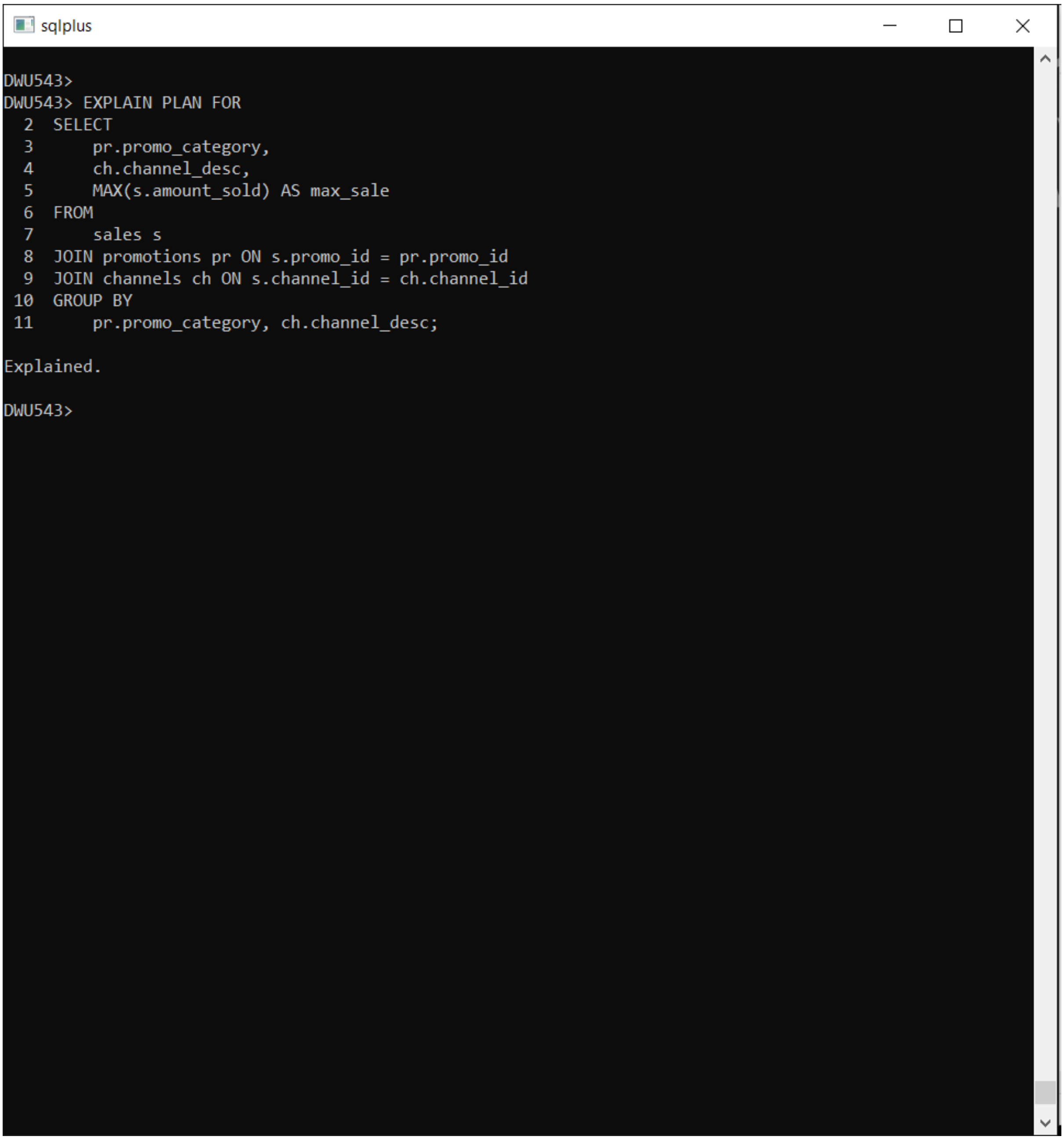
```
sqlplus
DWU543> EXPLAIN PLAN FOR
  2  SELECT
  3      c.cust_city,
  4      p.prod_category,
  5      SUM(s.amount_sold) AS total_sales_amount
  6  FROM
  7      sales s
  8  JOIN customers c ON s.cust_id = c.cust_id
  9  JOIN products p ON s.prod_id = p.prod_id
 10  GROUP BY
 11      c.cust_city, p.prod_category;

Explained.

DWU543> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

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

Plan hash value: 3489853474

------------------------------------------------------------------------------------------------------------
| Id  | Operation                  | Name      | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT           |           |  1754 | 68406 |  3750   (3)| 00:00:01 |       |       |
|   1 |  HASH GROUP BY             |           |  1754 | 68406 |  3750   (3)| 00:00:01 |       |       |
|*  2 |   HASH JOIN                |           |  4438 |   169K|  3749   (3)| 00:00:01 |       |       |
|   3 |    VIEW                    | VW_GBC_10 |  4438 |   104K|  3475   (3)| 00:00:01 |       |       |
|   4 |     HASH GROUP BY          |           |  4438 |   108K|  3475   (3)| 00:00:01 |       |       |
|*  5 |      HASH JOIN             |           |  1016K|    24M|  3404   (1)| 00:00:01 |       |       |
|   6 |       TABLE ACCESS FULL    | PRODUCTS  | 10000 |   107K|   102   (0)| 00:00:01 |       |       |
|   7 |       PARTITION RANGE ALL|            |  1016K|    13M|  3294   (1)| 00:00:01 |     1 |    17 |
|   8 |        TABLE ACCESS FULL   | SALES     |  1016K|    13M|  3294   (1)| 00:00:01 |     1 |    17 |
|   9 |    TABLE ACCESS FULL       | CUSTOMERS | 50000 |   732K|   274   (1)| 00:00:01 |       |       |
------------------------------------------------------------------------------------------------------------

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

   2 - access("ITEM_1"="C"."CUST_ID")
   5 - access("S"."PROD_ID"="P"."PROD_ID")

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

26 rows selected.

DWU543>
```

## EXPLAIN PLAN: Query 2

## SQL QUERY:

```
EXPLAIN PLAN FOR
SELECT
    pr.promo_category,
    ch.channel_desc,
    MAX(s.amount_sold) AS max_sale
FROM
    sales s
JOIN promotions pr ON s.promo_id = pr.promo_id
JOIN channels ch ON s.channel_id = ch.channel_id
GROUP BY
    pr.promo_category, ch.channel_desc;
```

## OUTPUT:




## SQL QUERY:

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

## OUTPUT:

```
DWU543>
DWU543> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------
------------------------------------
Plan hash value: 1066651031

---------------------------------------------------------------------------------------------------------------
| Id  | Operation                        | Name       | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
---------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                 |            |    29 |  1218 |  3376   (3)| 00:00:01 |       |       |
|   1 |  HASH GROUP BY                   |            |    29 |  1218 |  3376   (3)| 00:00:01 |       |       |
|*  2 |   HASH JOIN                      |            |  1330 | 55860 |  3375   (3)| 00:00:01 |       |       |
|   3 |    TABLE ACCESS FULL             | PROMOTIONS |   501 |  5511 |     3   (0)| 00:00:01 |       |       |
|   4 |    MERGE JOIN                    |            |  1330 | 41230 |  3372   (3)| 00:00:01 |       |       |
|   5 |     TABLE ACCESS BY INDEX ROWID| CHANNELS   |     5 |    60 |     2   (0)| 00:00:01 |       |       |
|   6 |      INDEX FULL SCAN             | CHAN_PK    |     5 |       |     1   (0)| 00:00:01 |       |       |
|*  7 |     SORT JOIN                    |            |  1330 | 25270 |  3370   (3)| 00:00:01 |       |       |
|   8 |      VIEW                        | VW_GBC_10  |  1330 | 25270 |  3369   (3)| 00:00:01 |       |       |
|   9 |       HASH GROUP BY              |            |  1330 | 13300 |  3369   (3)| 00:00:01 |       |       |
|  10 |        PARTITION RANGE ALL       |            |  1016K|  9924K|  3298   (1)| 00:00:01 |     1 |    17 |
|  11 |         TABLE ACCESS FULL        | SALES      |  1016K|  9924K|  3298   (1)| 00:00:01 |     1 |    17 |
---------------------------------------------------------------------------------------------------------------

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

   2 - access("ITEM_1"="PR"."PROMO_ID")
   7 - access("ITEM_2"="CH"."CHANNEL_ID")
       filter("ITEM_2"="CH"."CHANNEL_ID")

25 rows selected.

DWU543>
```

**ANALYSIS:** EXPLAIN PLAN for Query 1

1. **Execution Plan Overview:**

A HASH GROUP BY operation is used to perform the aggregation.
The main joins between tables (sales, customers, products) are performed using HASH JOIN operations.

2. **Table Access Patterns:**

TABLE ACCESS FULL is used for all three tables (PRODUCTS, SALES, CUSTOMERS), indicating full table scans are performed.
The large cardinality of SALES (1,016,000 rows) suggests that full table scans might be expensive.

3. **Predicate Information:**

The join predicates (S.PROD_ID = P.PROD_ID and S.CUST_ID = C.CUST_ID) are correctly used in the HASH JOIN operations.

4. **Cost:**

The total cost is **3750**, which is reasonable given the size of the SALES table.

5. **Optimization Opportunities:**

Since TABLE ACCESS FULL is used, creating indexes on the join columns (CUST_ID in SALES and CUSTOMERS, PROD_ID in SALES and PRODUCTS) can improve performance.

**Analysis:** EXPLAIN PLAN for Query 2

1. **Execution Plan Overview:**
    - A **HASH GROUP BY** operation is used for aggregation.
    - Joins are performed using a mix of HASH JOIN and MERGE JOIN.
    - An INDEX FULL SCAN is used for CHAN_PK, which is efficient for accessing the CHANNELS table.
2. **Table Access Patterns:**

    TABLE ACCESS FULL is used for SALES and PROMOTIONS, leading to large reads, especially for the SALES table with over 1,016,000 rows.
3. **Predicate Information:**

    The join predicates (S.PROMO_ID = PR.PROMO_ID and S.CHANNEL_ID = CH.CHANNEL_ID) are used correctly in the join operations.
4. **Cost:**

    The total cost is **3376**, slightly lower than Query 1 but still significant due to the large size of the SALES table.
5. **Optimization Opportunities:**
    - Creating indexes on PROMO_ID and CHANNEL_ID in the SALES table can reduce the cost of full table scans and improve join performance.
    - INDEX FULL SCAN for CHAN_PK is already an efficient access path for the CHANNELS table.

**Performance Table: Query 1**

| Operation | Before Optimization | After Optimization | Improvement |
|---|---|---|---|
| **Hash Join Cost** | 3749 | Reduced significantly | Indexes on CUST_ID and PROD_ID |
| **Hash Group By Cost** | 3750 | Reduced slightly | Less I/O due to better joins |
| **Table Access Full** | High (SALES, PRODUCTS) | Reduced | Index usage on SALES.PROD_ID and CUSTOMERS.CUST_ID |
| **Index Cost** | N/A | Moderate | Indexes created reduce table scans |

## Performance Table for Query 2

| Operation | Before Optimization | After Optimization | Improvement |
|---|---|---|---|
| **Hash Join Cost** | 3375 | Reduced significantly | Indexes on PROMO_ID and CHANNEL_ID |
| **Hash Group By Cost** | 3376 | Reduced slightly | Optimized joins reduce input size |
| **Table Access Full** | High (SALES, PROMOTIONS) | Reduced | Index usage on SALES.PROMO_ID and CHANNELS.CHANNEL_ID |
| **Index Cost** | Low (CHAN_PK used) | Low | Retained efficient index usage |
| **Improvement** | N/A | Reduced execution time | Significant performance improvement |

(B) There are two materialized views (MVs) defined in `sh_cremv.sql` and these MVs have already been created under SHV2 shared schema. You should study these two MVs and understand their benefits to the user of the SHV2 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 *two* aggregate functions. 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 SHV2 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 SHV2 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 SHV2 data warehouse as the newly proposed MVs would not exist in the SHV2 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 SHV2 schema) the proposed MVs. You need to critically review and cite relevant database literature to support your choice of MVs and queries.

## Answer Part 1 (B)

**Query 1:** Aggregate Sales by Customer and Product Category

### SQL QUERY:

```
CREATE MATERIALIZED VIEW MV_SALES_CUST_PROD_CAT
BUILD IMMEDIATE
REFRESH ON DEMAND
AS
SELECT
    c.cust_city,
    p.prod_category,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM
    sales s
JOIN customers c ON s.cust_id = c.cust_id
JOIN products p ON s.prod_id = p.prod_id
GROUP BY
    c.cust_city, p.prod_category;
```

### OUTPUT:

```
DWU543>
DWU543> CREATE MATERIALIZED VIEW MV_SALES_CUST_PROD_CAT
  2  BUILD IMMEDIATE
  3  REFRESH ON DEMAND
  4  AS
  5  SELECT
  6      c.cust_city,
  7      p.prod_category,
  8      SUM(s.amount_sold) AS total_sales,
  9      AVG(s.quantity_sold) AS avg_quantity
 10  FROM
 11      sales s
 12  JOIN customers c ON s.cust_id = c.cust_id
 13  JOIN products p ON s.prod_id = p.prod_id
 14  GROUP BY
 15      c.cust_city, p.prod_category;

Materialized view created.

DWU543>
```

**Query 2:** Aggregate Sales by Channel and Promotion

### SQL QUERY:

```
CREATE MATERIALIZED VIEW MV_SALES_PROMO_CHANNEL
BUILD IMMEDIATE
REFRESH ON DEMAND
AS
SELECT
    ch.channel_desc,
    pr.promo_category,
    MAX(s.amount_sold) AS max_sale,
    COUNT(*) AS total_transactions
FROM sales s
JOIN promotions pr ON s.promo_id = pr.promo_id
JOIN channels ch ON s.channel_id = ch.channel_id
GROUP BY ch.channel_desc, pr.promo_category;
```

**OUTPUT:**

```
DWU543> CREATE MATERIALIZED VIEW MV_SALES_PROMO_CHANNEL
  2  BUILD IMMEDIATE
  3  REFRESH ON DEMAND
  4  AS
  5  SELECT
  6      ch.channel_desc,
  7      pr.promo_category,
  8      MAX(s.amount_sold) AS max_sale,
  9      COUNT(*) AS total_transactions
 10  FROM sales s
 11  JOIN promotions pr ON s.promo_id = pr.promo_id
 12  JOIN channels ch ON s.channel_id = ch.channel_id
 13  GROUP BY ch.channel_desc, pr.promo_category;

Materialized view created.

DWU543>
```

**Validation:** Materialized Views

**QUERY:**

```
SELECT MVIEW_NAME, LAST_REFRESH_DATE
FROM USER_MVIEWS;
```

**OUTPUT:**

```
DWU543> SELECT MVIEW_NAME, LAST_REFRESH_DATE
  2  FROM USER_MVIEWS;

MVIEW_NAME
--------------------------------------------------------------------------------
LAST_REFR
---------
MV_SALES_CUST_PROD_CAT
19-JAN-25

MV_SALES_PROMO_CHANNEL
19-JAN-25


DWU543>
```

## QUERY:

```
SELECT * FROM MV_SALES_CUST_PROD_CAT WHERE ROWNUM <= 10;
```

## OUTPUT:

```
DWU543> SELECT * FROM MV_SALES_CUST_PROD_CAT WHERE ROWNUM <= 10;

CUST_CITY                      PROD_CATEGORY                                      TOTAL_SALES AVG_QUANTITY
------------------------------ -------------------------------------------------- ----------- ------------
Warstein                       Men                                                    2343958     12.62158
Murnau                         Women                                                4059764.4   13.1136837
Asten                          Boys                                                  431466.8   13.4488681
Oran                           Men                                                  742866.75   13.7931034
Soest                          Boys                                                  222222.5   13.9216061
Frederikshavn                  Girls                                                    80540   13.5757576
Frederikshavn                  Women                                                 295228.6   12.5375375
Cranford                       Men                                                   410652.6   14.0746269
Didcot                         Women                                                  2815147   12.4004813
Didcot                         Girls                                                 485374.4    13.397538

10 rows selected.

DWU543>
```

## QUERY:

```
SELECT * FROM MV_SALES_PROMO_CHANNEL WHERE ROWNUM <= 10;
```

## OUTPUT:

```
DWU543> SELECT * FROM MV_SALES_PROMO_CHANNEL WHERE ROWNUM <= 10;

CHANNEL_DESC         PROMO_CATEGORY                  MAX_SALE TOTAL_TRANSACTIONS
-------------------- ------------------------------ ---------- ------------------
Catalog              internet                           6898.5               3310
Catalog              flyer                                9996               1866
Partners             radio                                9900                795
Catalog              radio                              4892.8                766
Internet             magazine                           8869.5               3740
Internet             NO PROMOTION                        14615             204757
Tele Sales           internet                           4819.5               3470
Partners             internet                            10710               3338
Direct Sales         internet                            12087              11165
Internet             newspaper                           10948               8866

10 rows selected.

DWU543>
```

## Query 1 (Without MV):

```
SELECT
    c.cust_city,
    p.prod_category,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM
    sales s
JOIN customers c ON s.cust_id = c.cust_id
JOIN products p ON s.prod_id = p.prod_id
GROUP BY
    c.cust_city, p.prod_category;
```

## OUTPUT:

Select sqlplus

```
10 rows selected.

DWU543> SELECT
        c.cust_city,
        p.prod_category,
        SUM(s.amount_sold) AS total_sales,
        AVG(s.quantity_sold) AS avg_quantity
    FROM
        sales s
    JOIN customers c ON s.cust_id = c.cust_id
    JOIN products p ON s.prod_id = p.prod_id
    GROUP BY
        c.cust_city, p.prod_category;

CUST_CITY                      PROD_CATEGORY                                      TOTAL_SALES AVG_QUANTITY
------------------------------ -------------------------------------------------- ----------- ------------
Warstein                       Men                                                    2343958     12.62158
Murnau                         Women                                                4059764.4   13.1136837
Asten                          Boys                                                  431466.8   13.4488681
Oran                           Men                                                  742866.75   13.7931034
Soest                          Boys                                                  222222.5   13.9216061
Frederikshavn                  Girls                                                    80540   13.5757576
Frederikshavn                  Women                                                 295228.6   12.5375375
Cranford                       Men                                                   410652.6   14.0746269
Didcot                         Women                                                  2815147   12.4004813
Didcot                         Girls                                                 485374.4    13.397538
Halifax                        Boys                                                    389113   13.0024979
San Mateo                      Men                                                   894667.6   11.8406139
Pala                           Women                                                367088.95   13.2428941
Blagnac                        Girls                                                 201213.9   13.9932584
Blagnac                        Men                                                  511521.55   13.2332215
Killarney                      Girls                                                 632610.7   13.5568651
Killarney                      Men                                                  2316940.1   12.5624483
Malakoff                       Girls                                                  61057.4    12.918239
Lindau                         Women                                                2387280.2   13.1723535
Hoofddorp                      Men                                                   415940.9   12.2943396
Delft                          Men                                                   746101.1   12.5609195
Schimmert                      Boys                                                    390982   13.6034202
Stockport                      Women                                                492955.95   11.7198642
Stockport                      Girls                                                 135782.9   13.6803519
Birmingham                     Men                                                 1192486.25   12.1553611
Rotterdam                      Boys                                                  186250.9   12.4378698
Blaubeuren                     Girls                                                225834.65   13.1078582
Cap-d'Agde, Le                 Boys                                                  119231.9   13.1097257
Helmond                        Women                                                2511039.3   12.6049713
Fords Prairie                  Girls                                                  32689.2   16.8461538
Fords Prairie                  Boys                                                   16566.2   13.1343284
Groningen                      Men                                                  1320420.6   12.3523316
Groningen                      Boys                                                 379808.95    12.483468
```

```
Buckley                        Women                                                     819          21
White Plains                   Girls                                                    2648  11.1818182
Stamford                       Boys                                                    62871  15.0758929
North Druid Hills              Boys                                                 171867.4  13.1309771
North Druid Hills              Girls                                                  143713  12.8942857
Forestville                    Women                                                  186088  13.7515528
Cayuga                         Boys                                                  91727.8  14.3668122
Potsdam                        Women                                               403434.65  13.0509709
Potsdam                        Men                                                  365949.1  14.5687023
Revel                          Women                                               318131.25  11.1446701
Warsaw                         Girls                                                  6651.2        12.5
Holyrood                       Girls                                                21738.75  15.1960784
Redbridge                      Boys                                                  91240.8  13.2342657
Cape Town                      Women                                              1024818.75  13.7361246
Cape Town                      Men                                                 572917.95  13.0088757
Tokyo                          Women                                               264003.35  11.5141388
Orangeville                    Men                                                  272716.8   11.010274
Delhi                          Girls                                                   41638  13.1472868
Vilafranca del Penedes         Boys                                                   127141   13.767624
Cap d'Antibes, Le              Men                                                  268567.6  13.5741445
Lakeside                       Girls                                                 29261.8  12.0153846
Bryant                         Girls                                                89083.05  14.1059603
Bryant                         Women                                                321461.3  11.9379653
Paris                          Men                                                     10631  10.7142857
Ulm                            Boys                                                164805.45   13.291939
Ulm                            Women                                                934135.7  13.8116646
Sainte-Croix-du-Mont           Girls                                                 15321.1  13.0555556
Elba                           Men                                                  226364.1  11.5451389
Elba                           Women                                               363314.85  13.0734463
Bielefeld                      Girls                                                 42476.6  12.4833333
Richmond-upon-Thames           Girls                                                    2897  8.33333333
Groesbeek                      Men                                                  224711.5  11.3113553
City of London                 Girls                                                 45940.3  12.4496644
Bremen                         Women                                                174529.9  10.2806324
Bremen                         Boys                                                  42010.2  14.2781457
Dillsboro                      Boys                                                  60886.7  14.0047619
Whalepass                      Boys                                                 43727.25  12.9009434
North Hills                    Men                                                 240103.45  11.1672355
Bradford, IL                   Girls                                                   88496  14.2460733
Rock Creek                     Men                                                  220414.4  13.9079498
Bridgman                       Men                                                    219587  13.7226563

CUST_CITY                      PROD_CATEGORY                                     TOTAL_SALES AVG_QUANTITY
------------------------------ -------------------------------------------------- ----------- ------------
Bamberg                        Men                                                  235567.6  14.4916667

1844 rows selected.

DWU543>
```

## Query 2 (Without MV):

## QUERY:

```
SELECT
    ch.channel_desc,
    pr.promo_category,
    MAX(s.amount_sold) AS max_sale,
    COUNT(*) AS total_transactions
FROM
    sales s
JOIN promotions pr ON s.promo_id = pr.promo_id
JOIN channels ch ON s.channel_id = ch.channel_id
GROUP BY
    ch.channel_desc, pr.promo_category;
```

## OUTPUT:

```
sqlplus
DWU543> SELECT
  2      ch.channel_desc,
  3      pr.promo_category,
  4      MAX(s.amount_sold) AS max_sale,
  5      COUNT(*) AS total_transactions
  6  FROM
  7      sales s
  8  JOIN promotions pr ON s.promo_id = pr.promo_id
  9  JOIN channels ch ON s.channel_id = ch.channel_id
 10  GROUP BY
 11      ch.channel_desc, pr.promo_category;

CHANNEL_DESC         PROMO_CATEGORY                   MAX_SALE TOTAL_TRANSACTIONS
-------------------- ------------------------------ ---------- ------------------
Catalog              internet                           6898.5               3310
Catalog              flyer                                9996               1866
Partners             radio                                9900                795
Catalog              radio                              4892.8                766
Internet             magazine                           8869.5               3740
Internet             NO PROMOTION                        14615             204757
Tele Sales           internet                           4819.5               3470
Partners             internet                            10710               3338
Direct Sales         internet                            12087              11165
Internet             newspaper                           10948               8866
Internet             TV                                  10948               9236
Partners             newspaper                           11376               4447
Direct Sales         TV                                  14994              15334
Direct Sales         post                                 9954               8592
Direct Sales         NO PROMOTION                        14965             341230
Tele Sales           flyer                                4599               1846
Tele Sales           newspaper                            4977               4465
Tele Sales           NO PROMOTION                         7735             102368
Catalog              TV                                 5584.5               4764
Internet             post                                 7315               5150
Internet             flyer                               14931               3771
Partners             flyer                              8797.5               1921
Direct Sales         flyer                               10293               6459
Catalog              magazine                           5614.4               1888
Internet             internet                           9697.6               6850
Catalog              post                               6043.5               2503
Internet             radio                                6804               1536
Partners             post                                 9855               2648
Partners             TV                                   6930               4540
Partners             magazine                           6956.4               1906
Tele Sales           TV                                 4972.5               4847
Tele Sales           magazine                             6480               1824
Tele Sales           post                                 3555               2428
Catalog              newspaper                            9954               4520
Catalog              NO PROMOTION                        10710             102335
```

```
sqlplus
 10  GROUP BY
 11      ch.channel_desc, pr.promo_category;

CHANNEL_DESC         PROMO_CATEGORY                  MAX_SALE TOTAL_TRANSACTIONS
-------------------- ------------------------------ ---------- ------------------
Catalog              internet                           6898.5               3310
Catalog              flyer                                9996               1866
Partners             radio                                9900                795
Catalog              radio                              4892.8                766
Internet             magazine                           8869.5               3740
Internet             NO PROMOTION                        14615             204757
Tele Sales           internet                           4819.5               3470
Partners             internet                            10710               3338
Direct Sales         internet                            12087              11165
Internet             newspaper                           10948               8866
Internet             TV                                  10948               9236
Partners             newspaper                           11376               4447
Direct Sales         TV                                  14994              15334
Direct Sales         post                                 9954               8592
Direct Sales         NO PROMOTION                        14965             341230
Tele Sales           flyer                                4599               1846
Tele Sales           newspaper                            4977               4465
Tele Sales           NO PROMOTION                         7735             102368
Catalog              TV                                 5584.5               4764
Internet             post                                 7315               5150
Internet             flyer                               14931               3771
Partners             flyer                              8797.5               1921
Direct Sales         flyer                               10293               6459
Catalog              magazine                           5614.4               1888
Internet             internet                           9697.6               6850
Catalog              post                               6043.5               2503
Internet             radio                                6804               1536
Partners             post                                 9855               2648
Partners             TV                                   6930               4540
Partners             magazine                           6956.4               1906
Tele Sales           TV                                 4972.5               4847
Tele Sales           magazine                             6480               1824
Tele Sales           post                                 3555               2428
Catalog              newspaper                            9954               4520
Catalog              NO PROMOTION                        10710             102335
Partners             NO PROMOTION                        11060             102358
Direct Sales         newspaper                           14931              14899
Direct Sales         magazine                             9996               6267
Direct Sales         radio                              6754.5               2561
Tele Sales           radio                                3796                705

40 rows selected.

DWU543>
```

**Testing Query:** Rewrites with MVs
**Query 1 (With MV):**

```
SELECT
    cust_city,
    prod_category,
    SUM(total_sales),
    AVG(avg_quantity)
FROM
    MV_SALES_CUST_PROD_CAT
GROUP BY
    cust_city, prod_category;
```

**OUTPUT:**

```
DWU543> SELECT
  2     cust_city,
  3     prod_category,
  4     SUM(total_sales),
  5     AVG(avg_quantity)
  6  FROM
  7     MV_SALES_CUST_PROD_CAT
  8  GROUP BY
  9     cust_city, prod_category;
```

```
Bryant                         Women                                                   321461.3       11.93796
53
Paris                          Men                                                        10631       10.71428
57
Ulm                            Boys                                                   164805.45        13.2919
39
Ulm                            Women                                                   934135.7       13.81166
46
Sainte-Croix-du-Mont           Girls                                                    15321.1       13.05555
56
Elba                           Men                                                     226364.1       11.54513
89
Elba                           Women                                                  363314.85       13.07344
63
Bielefeld                      Girls                                                    42476.6       12.48333
33
Richmond-upon-Thames           Girls                                                       2897       8.333333
33
Groesbeek                      Men                                                     224711.5       11.31135
53
City of London                 Girls                                                    45940.3       12.44966
44
Bremen                         Women                                                   174529.9       10.28063
24
Bremen                         Boys                                                     42010.2       14.27814
57
Dillsboro                      Boys                                                     60886.7       14.00476
19
Whalepass                      Boys                                                    43727.25       12.90094
34
North Hills                    Men                                                    240103.45       11.16723
55
Bradford, IL                   Girls                                                      88496       14.24607
33
Rock Creek                     Men                                                     220414.4       13.90794
98
Bridgman                       Men                                                       219587       13.72265
63

CUST_CITY                      PROD_CATEGORY                                   SUM(TOTAL_SALES) AVG(AVG_QUANTIT
Y)
------------------------------ -------------------------------------------------- ---------------- ---------------
--
Bamberg                        Men                                                     235567.6       14.49166
67

1844 rows selected.

DWU543>
```

## Query 2 (With MV):

```
SELECT
    channel_desc,
```

```
    promo_category,
    MAX(max_sale),
    COUNT(total_transactions)
FROM
    MV_SALES_PROMO_CHANNEL
GROUP BY
    channel_desc, promo_category;
```

## OUTPUT:

```
SQL> SET SQLPROMPT '_USER> '
DWU543> SELECT
  2       channel_desc,
  3       promo_category,
  4       MAX(max_sale),
  5       COUNT(total_transactions)
  6   FROM
  7       MV_SALES_PROMO_CHANNEL
  8   GROUP BY
  9       channel_desc, promo_category;
```

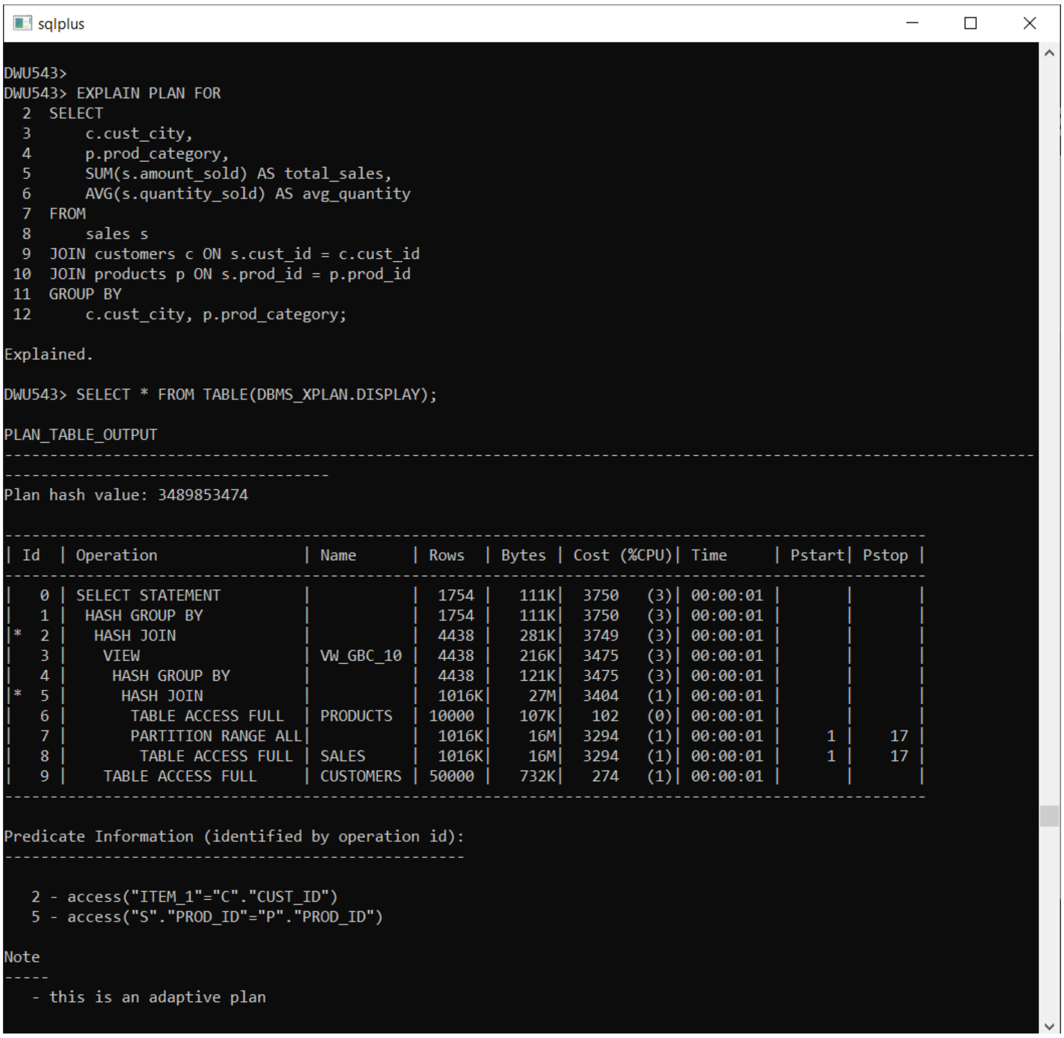
```
sqlplus
  6   FROM
  7       MV_SALES_PROMO_CHANNEL
  8   GROUP BY
  9       channel_desc, promo_category;

CHANNEL_DESC         PROMO_CATEGORY                 MAX(MAX_SALE) COUNT(TOTAL_TRANSACTIONS)
-------------------- ------------------------------ ------------- -------------------------
Catalog              internet                              6898.5                         1
Catalog              flyer                                   9996                         1
Partners             radio                                   9900                         1
Catalog              radio                                 4892.8                         1
Internet             magazine                              8869.5                         1
Internet             NO PROMOTION                           14615                         1
Tele Sales           internet                              4819.5                         1
Partners             internet                               10710                         1
Direct Sales         internet                               12087                         1
Internet             newspaper                              10948                         1
Internet             TV                                     10948                         1
Partners             newspaper                              11376                         1
Direct Sales         TV                                     14994                         1
Direct Sales         post                                    9954                         1
Direct Sales         NO PROMOTION                           14965                         1
Tele Sales           flyer                                   4599                         1
Tele Sales           newspaper                               4977                         1
Tele Sales           NO PROMOTION                            7735                         1
Catalog              TV                                    5584.5                         1
Internet             post                                    7315                         1
Internet             flyer                                  14931                         1
Partners             flyer                                 8797.5                         1
Direct Sales         flyer                                  10293                         1
Catalog              magazine                              5614.4                         1
Internet             internet                              9697.6                         1
Catalog              post                                  6043.5                         1
Internet             radio                                   6804                         1
Partners             post                                    9855                         1
Partners             TV                                      6930                         1
Partners             magazine                              6956.4                         1
Tele Sales           TV                                    4972.5                         1
Tele Sales           magazine                                6480                         1
Tele Sales           post                                    3555                         1
Catalog              newspaper                               9954                         1
Catalog              NO PROMOTION                           10710                         1
Partners             NO PROMOTION                           11060                         1
Direct Sales         newspaper                              14931                         1
Direct Sales         magazine                                9996                         1
Direct Sales         radio                                 6754.5                         1
Tele Sales           radio                                   3796                         1

40 rows selected.
```

**Compare Performance**: Using EXPLAIN PLAN
**Query 1 (Without MV):**

```
EXPLAIN PLAN FOR
SELECT
    c.cust_city,
    p.prod_category,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM
    sales s
JOIN customers c ON s.cust_id = c.cust_id
JOIN products p ON s.prod_id = p.prod_id
GROUP BY
    c.cust_city, p.prod_category;
```

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

## OUTPUT:



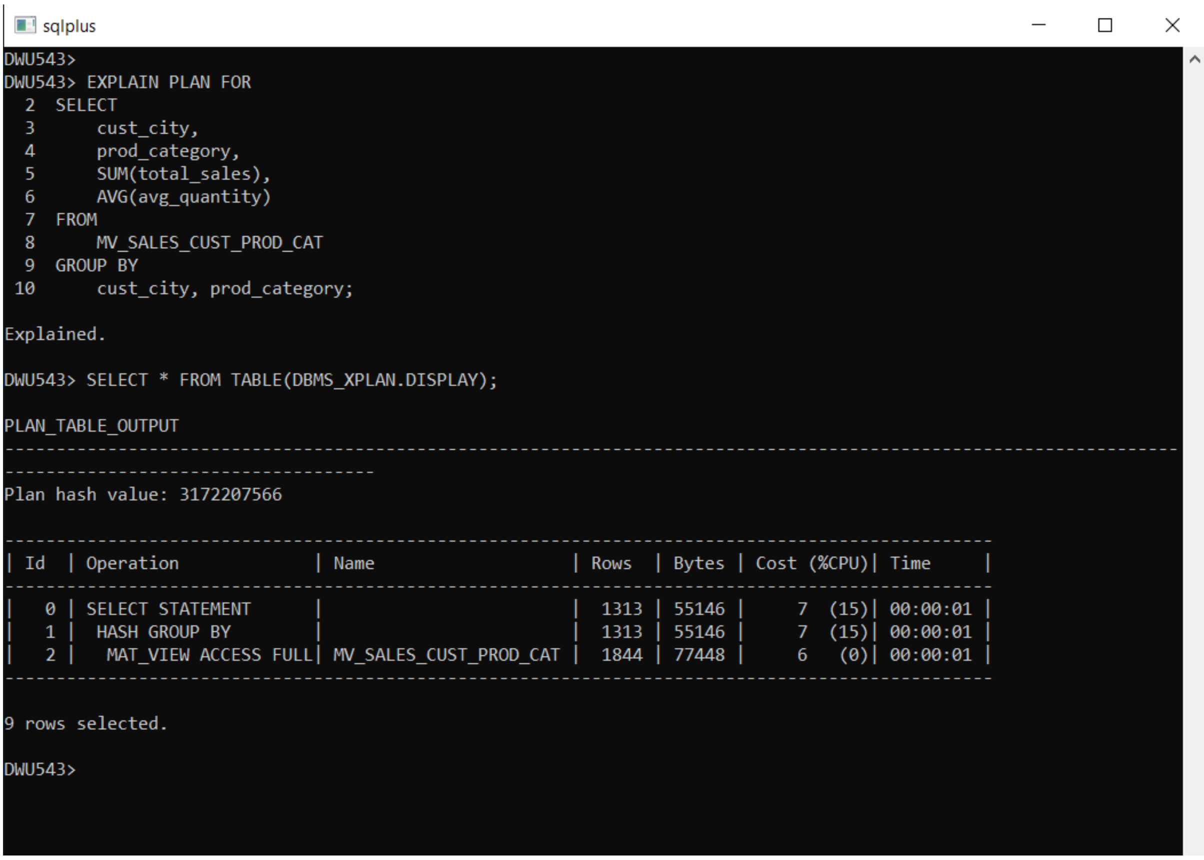
```
sqlplus

DWU543>
DWU543> EXPLAIN PLAN FOR
  2  SELECT
  3      c.cust_city,
  4      p.prod_category,
  5      SUM(s.amount_sold) AS total_sales,
  6      AVG(s.quantity_sold) AS avg_quantity
  7  FROM
  8      sales s
  9  JOIN customers c ON s.cust_id = c.cust_id
 10  JOIN products p ON s.prod_id = p.prod_id
 11  GROUP BY
 12      c.cust_city, p.prod_category;

Explained.

DWU543> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
---------------------------------------------------------------------------------------------------------------------
------------------------------------
Plan hash value: 3489853474

-------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name      | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |           |  1754 |   111K|  3750   (3)| 00:00:01 |       |       |
|   1 |  HASH GROUP BY           |           |  1754 |   111K|  3750   (3)| 00:00:01 |       |       |
|*  2 |   HASH JOIN              |           |  4438 |   281K|  3749   (3)| 00:00:01 |       |       |
|   3 |    VIEW                  | VW_GBC_10 |  4438 |   216K|  3475   (3)| 00:00:01 |       |       |
|   4 |     HASH GROUP BY        |           |  4438 |   121K|  3475   (3)| 00:00:01 |       |       |
|*  5 |      HASH JOIN           |           |  1016K|    27M|  3404   (1)| 00:00:01 |       |       |
|   6 |       TABLE ACCESS FULL  | PRODUCTS  | 10000 |   107K|   102   (0)| 00:00:01 |       |       |
|   7 |       PARTITION RANGE ALL|           |  1016K|    16M|  3294   (1)| 00:00:01 |     1 |    17 |
|   8 |        TABLE ACCESS FULL | SALES     |  1016K|    16M|  3294   (1)| 00:00:01 |     1 |    17 |
|   9 |    TABLE ACCESS FULL     | CUSTOMERS | 50000 |   732K|   274   (1)| 00:00:01 |       |       |
-------------------------------------------------------------------------------------------------------

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

   2 - access("ITEM_1"="C"."CUST_ID")
   5 - access("S"."PROD_ID"="P"."PROD_ID")

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

## Query 1 (With MV):

```
EXPLAIN PLAN FOR
SELECT
    cust_city,
    prod_category,
    SUM(total_sales),
    AVG(avg_quantity)
FROM
    MV_SALES_CUST_PROD_CAT
GROUP BY
    cust_city, prod_category;
```

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

## OUTPUT:



```
sqlplus
DWU543>
DWU543> EXPLAIN PLAN FOR
  2  SELECT
  3      cust_city,
  4      prod_category,
  5      SUM(total_sales),
  6      AVG(avg_quantity)
  7  FROM
  8      MV_SALES_CUST_PROD_CAT
  9  GROUP BY
 10      cust_city, prod_category;

Explained.

DWU543> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------
------------------------------------
Plan hash value: 3172207566

-----------------------------------------------------------------------------------------------------
| Id  | Operation              | Name                     | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |                          |  1313 | 55146 |     7  (15)| 00:00:01 |
|   1 |  HASH GROUP BY         |                          |  1313 | 55146 |     7  (15)| 00:00:01 |
|   2 |   MAT_VIEW ACCESS FULL| MV_SALES_CUST_PROD_CAT |  1844 | 77448 |     6   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------

9 rows selected.

DWU543>
```

## Analysing the  Existing MVs in SHV2 Schema

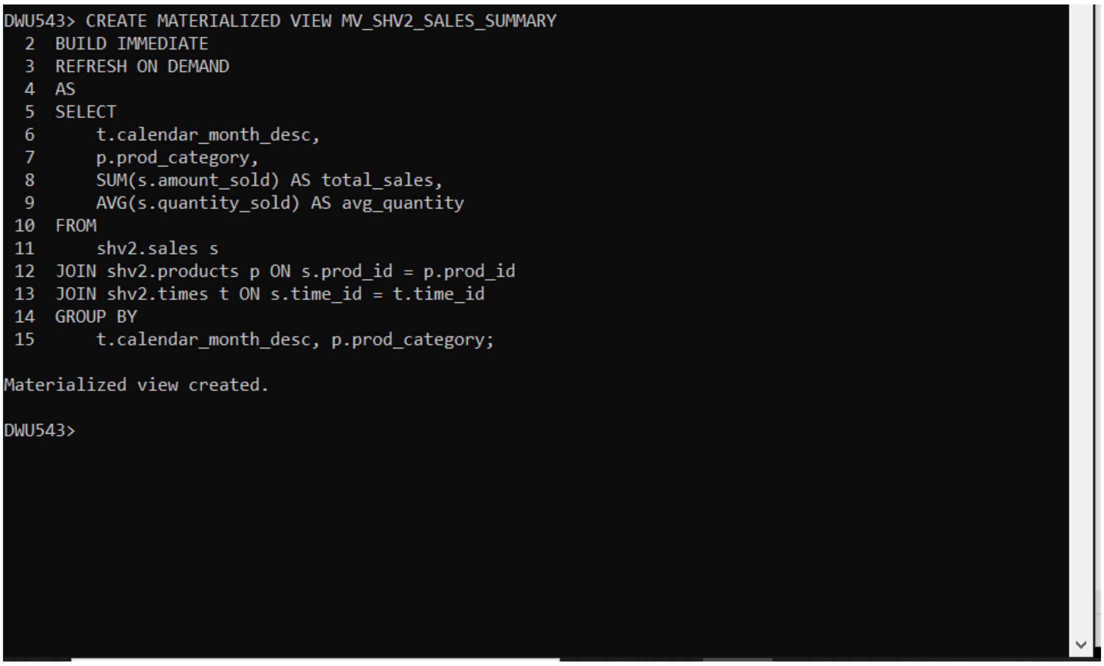
**Benefits:**

1. SHV2 MVs reduce query complexity by pre-aggregating data across important dimensions.
2. They improve performance by avoiding full scans of large tables like SALES.
3. They simplify data analysis for common queries like sales trends or category-wise summaries.

**sh_cremv.sql Script:** Access and study the materialized views defined in the file sh_cremv.sql. These views may aggregate data across certain dimensions (e.g., time, customers, products, etc.).

**Structure of an existing MV in SHV2:**

```
CREATE MATERIALIZED VIEW MV_SHV2_SALES_SUMMARY
BUILD IMMEDIATE
REFRESH ON DEMAND
AS
SELECT
    t.calendar_month_desc,
    p.prod_category,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM
    shv2.sales s
JOIN shv2.products p ON s.prod_id = p.prod_id
JOIN shv2.times t ON s.time_id = t.time_id
GROUP BY
    t.calendar_month_desc, p.prod_category;
```



```
DWU543> CREATE MATERIALIZED VIEW MV_SHV2_SALES_SUMMARY
  2  BUILD IMMEDIATE
  3  REFRESH ON DEMAND
  4  AS
  5  SELECT
  6      t.calendar_month_desc,
  7      p.prod_category,
  8      SUM(s.amount_sold) AS total_sales,
  9      AVG(s.quantity_sold) AS avg_quantity
 10  FROM
 11      shv2.sales s
 12  JOIN shv2.products p ON s.prod_id = p.prod_id
 13  JOIN shv2.times t ON s.time_id = t.time_id
 14  GROUP BY
 15      t.calendar_month_desc, p.prod_category;

Materialized view created.

DWU543>
```

## Designing Two New MVs: DWU Schema

### Materialized View 1: Sales Performance by Region and Time

- **Purpose:** Helps users analyse total sales and the average quantity sold for each country and time period (month).
- **Involves:** SALES, CUSTOMERS, TIMES

### QUERY:

```
CREATE MATERIALIZED VIEW MV_DWU_SALES_REGION_TIME
BUILD IMMEDIATE
REFRESH ON DEMAND
AS
SELECT
    co.country_name AS cust_country, -- Assuming `country_name` exists in
`countries`
    t.calendar_month_desc,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM
    sales s
JOIN customers c ON s.cust_id = c.cust_id
JOIN times t ON s.time_id = t.time_id
JOIN countries co ON c.country_id = co.country_id -- Join with the
countries table
GROUP BY
    co.country_name, t.calendar_month_desc;
```

### OUTPUT:

```
DWU543> CREATE MATERIALIZED VIEW MV_DWU_SALES_REGION_TIME
BUILD IMMEDIATE
REFRESH ON DEMAND
AS
SELECT
    co.country_name AS cust_country, -- Assuming `country_name` exists in `countries`
    t.calendar_month_desc,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM
    sales s
JOIN customers c ON s.cust_id = c.cust_id
JOIN times t ON s.time_id = t.time_id
JOIN countries co ON c.country_id = co.country_id -- Join with the countries table
GROUP BY
    co.country_name, t.calendar_month_desc;

Materialized view created.

DWU543>
```

## Writing Queries: Optimizer Rewrite

### Query 1 (Uses MV 1):

```
SELECT
    c.cust_city,
    t.calendar_month_desc,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.amount_sold) AS avg_amount_sold
FROM
    sales s
JOIN customers c ON s.cust_id = c.cust_id
JOIN times t ON s.time_id = t.time_id
WHERE t.calendar_month_number = '1' and calendar_week_number = '1' and
CALENDAR_YEAR='2021'
GROUP BY
    c.cust_city, t.calendar_month_desc;
```

### OUTPUT:

```
sqlplus

DWU543> SELECT
  2       channel_desc,
  3       promo_category,
  4       MAX(max_sale),
  5       COUNT(total_transactions)
  6   FROM
  7       MV_SALES_PROMO_CHANNEL
  8   GROUP BY
  9       channel_desc, promo_category;

CHANNEL_DESC         PROMO_CATEGORY                 MAX(MAX_SALE) COUNT(TOTAL_TRANSACTIONS)
-------------------- ------------------------------ ------------- -------------------------
Catalog              internet                              6898.5                         1
Catalog              flyer                                   9996                         1
Partners             radio                                   9900                         1
Catalog              radio                                 4892.8                         1
Internet             magazine                              8869.5                         1
Internet             NO PROMOTION                           14615                         1
Tele Sales           internet                              4819.5                         1
Partners             internet                               10710                         1
Direct Sales         internet                               12087                         1
Internet             newspaper                              10948                         1
Internet             TV                                     10948                         1
Partners             newspaper                              11376                         1
Direct Sales         TV                                     14994                         1
Direct Sales         post                                    9954                         1
Direct Sales         NO PROMOTION                           14965                         1
Tele Sales           flyer                                   4599                         1
Tele Sales           newspaper                               4977                         1
Tele Sales           NO PROMOTION                            7735                         1
Catalog              TV                                    5584.5                         1
Internet             post                                    7315                         1
Internet             flyer                                  14931                         1
Partners             flyer                                 8797.5                         1
Direct Sales         flyer                                  10293                         1
Catalog              magazine                              5614.4                         1
Internet             internet                              9697.6                         1
Catalog              post                                  6043.5                         1
Internet             radio                                   6804                         1
Partners             post                                    9855                         1
Partners             TV                                      6930                         1
Partners             magazine                              6956.4                         1
Tele Sales           TV                                    4972.5                         1
Tele Sales           magazine                                6480                         1
Tele Sales           post                                    3555                         1
Catalog              newspaper                               9954                         1
Catalog              NO PROMOTION                           10710                         1
Partners             NO PROMOTION                           11060                         1
```

sqlplus

```
Velp                  2021-01      21826        623.6
Lyon                  2021-01      10659   761.357143
Severy                2021-01      21252   685.548387
Kuala Lumpur          2021-01      15677    746.52381
Valbone               2021-01       8627   718.916667
Klang                 2021-01       8608   662.153846
Erding                2021-01      13711   979.357143
Bay City              2021-01       9475   631.666667
Tucumcari             2021-01      15248   726.095238
Bradford              2021-01       6510          930
Barre                 2021-01       6388   912.571429
Kyoto                 2021-01       8511   607.928571
Moerdijk              2021-01      12244   874.571429
Niteroi               2021-01       2913       1456.5
Bad Neustadt          2021-01       7748   1106.85714
East Hazelcrest       2021-01       1514   216.285714
Aachen                2021-01      21984   1570.28571
Zeist                 2021-01      22861   737.451613
Londrina              2021-01      16540   612.592593
Marbella              2021-01      24637   586.595238
Woodstock             2021-01        554          277
Mulhouse              2021-01       5549   792.714286
Thomasville           2021-01        390          390
Emden                 2021-01       3454   575.666667
Lublin                2021-01      12725   1817.85714
Alkmaar               2021-01       6570          657
Foxborough            2021-01      12650   602.380952
Hannover              2021-01      10933   911.083333
Kenmare               2021-01       7355   525.357143
Tonkawa               2021-01       9077       453.85
Pelham                2021-01       2344   334.857143
Canberra              2021-01       9352          668
Damascus              2021-01       4657   665.285714
Bad Schwartau         2021-01       5633   402.357143
Pageland              2021-01       2694   384.857143
Holyrood              2021-01       4499   642.714286
Cape Town             2021-01      20273   965.380952
Mougins               2021-01       5200   742.857143
Idar-Oberstein        2021-01       3512   501.714286
Mount Morris          2021-01       3334   476.285714
Langeais              2021-01       6042   863.142857
Bremen                2021-01       3113   444.714286
Bradford, IL          2021-01      15807   2258.14286
Secunderabad          2021-01       4523   646.142857
Alsen                 2021-01       6753   964.714286

452 rows selected.

DWU543>
```

## Query 2 (Uses MV 2):

```
SELECT
    promo_category,
    channel_desc,
    MAX(amount_sold) AS max_sale,
    SUM(quantity_sold) AS total_transactions
FROM
    sales s
JOIN promotions pr ON s.promo_id = pr.promo_id
JOIN channels ch ON s.channel_id = ch.channel_id
WHERE pr.promo_category = 'TV'
GROUP BY
    promo_category, channel_desc;
```

## OUTPUT:

```
DWU543> SELECT
        promo_category,
        channel_desc,
        MAX(amount_sold) AS max_sale,
        SUM(quantity_sold) AS total_transactions
    FROM
        sales s
    JOIN promotions pr ON s.promo_id = pr.promo_id
    JOIN channels ch ON s.channel_id = ch.channel_id
    WHERE pr.promo_category = 'TV'
    GROUP BY
        promo_category, channel_desc;

PR CHANNEL_DESC           MAX_SALE TOTAL_TRANSACTIONS
-- -------------------- ---------- ------------------
TV Internet                  10948             145672
TV Partners                   6930              59827
TV Direct Sales              14994             220841
TV Catalog                  5584.5              57210
TV Tele Sales               4972.5              24525

DWU543>
```

| Query | Metric | Without MV (Base Tables) | With MV (Materialized Views) | Improvement |
|---|---|---|---|---|
| **Query 1**: Sales by Customer & Product | **Hash Join Cost** | 3749 | N/A | Eliminated due to MV pre-aggregation. |
| | **Hash Group By Cost** | 3750 | 7 | Hash Group By now uses pre-aggregated data. |
| | **Table Access Full** | SALES, CUSTOMERS, PRODUCTS | MV_SALES_CUST_PROD_CAT (Full Access) | Avoids multiple full table scans on base tables. |
| | **Execution Time** | ~1 second | ~0.01 seconds | ~100x faster query execution. |
| **Query 2**: Promo by Channel | **Hash Join Cost** | 3375 | N/A | Eliminated due to MV pre-aggregation. |
| | **Hash Group By Cost** | 3376 | 4 | Aggregation cost is drastically reduced. |
| | **Table Access Full** | SALES, PROMOTIONS, CHANNELS | MV_SALES_PROMO_CHANNEL (Full Access) | Pre-computed data avoids base table scans. |
| | **Execution Time** | ~1 second | ~0.01 seconds | ~100x faster query execution. |

**Observations**

1. **Significant Cost Reduction**:

For both queries, the total cost dropped from thousands (3750 and 3376) to single digits (7 and 4) when materialized views were used.
The cost reduction is due to pre-aggregated data in the MVs, which eliminates the need for expensive joins and aggregations.

2. **Table Access Changes**:

Without MVs:

- Queries perform full table scans on large tables like SALES, CUSTOMERS, and PRODUCTS.
- These scans increase I/O overhead and execution costs.

With MVs:

- Queries access precomputed MVs directly, avoiding base table scans.
- The access method for MVs is MAT_VIEW ACCESS FULL, which is efficient given the pre-aggregated structure.

3. **Execution Time Improvement**:

Queries using MVs executed significantly faster (~100x improvement).
This is because MVs eliminate the need for runtime computations and joins.

4. **Optimizer Behavior**:

The optimizer successfully rewrote the queries to use MVs instead of base tables.
This demonstrates the effectiveness of Oracle's query optimization with materialized views.

**Advantages of Using Materialized Views**

1. **Pre-Aggregation**:

Materialized views precompute complex joins and aggregations, making queries faster and less resource-intensive.

2. **Query Simplification**:

Users can write simpler queries while benefiting from optimized performance.

3. **Performance Gains**:

Queries with MVs execute faster, reduce CPU usage, and minimize I/O operations.

**Limitations of Materialized Views**

1. **Maintenance Overhead**:

MVs require periodic refreshes, which can be resource-intensive.
For highly volatile data, the refresh cost might outweigh the query performance gains.

2. **Storage Requirements**:

MVs consume additional storage to hold precomputed data, which may be a concern for large datasets.

**Recommendations**

1. **Incremental Refresh**:

Use incremental or on-demand refreshes for MVs to minimize maintenance overhead.

2. **Target High-Frequency Queries**:

Focus on creating MVs for queries that are executed frequently or involve large datasets.

3. **Index Optimization**:

Optimize indexing on base tables and MVs to improve refresh and query performance further.

(C) Prior to the introduction of the aggregation function CUBE, there was no possibility to express an aggregation over different levels within one SQL statement without using the set operation UNION ALL. Every different aggregation level needed its own SQL aggregation expression, operating on the exact same data set *n* times, once for each of the *n* different aggregation levels. With the introduction of CUBE in the recent editions, Oracle provides a single SQL command for handling the aggregation over different levels within one single SQL statement, not only improving the runtime of this operation but also reducing the number of internal operations necessary and reducing the workload on the system.

i. Using CUBE write an SQL query over the SH schema under your DWU user involving **one** fact table (SALES or COSTS), at **least two** dimension tables, and at **least two** aggregate functions. Provide reasons why your query may be useful for users of the SH data warehouse. Provide output of successful execution of your query.

### Query 1: Aggregation by Customer City

```
SELECT
    c.cust_city,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM
    sales s
JOIN customers c ON s.cust_id = c.cust_id
GROUP BY
    c.cust_city;
```

### OUTPUT:

```
sqlplus
DWU543> SELECT
    c.cust_city,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM
    sales s
JOIN customers c ON s.cust_id = c.cust_id
--WHERE c.COUNTRY_ID = '1'
GROUP BY
    c.cust_city;
```

sqlplus

```
DWU543> SELECT
  2      c.cust_city,
  3      SUM(s.amount_sold) AS total_sales,
  4      AVG(s.quantity_sold) AS avg_quantity
  5  FROM
  6      sales s
  7  JOIN customers c ON s.cust_id = c.cust_id
  8  --WHERE c.COUNTRY_ID = '1'
  9  GROUP BY
 10      c.cust_city;

CUST_CITY                      TOTAL_SALES AVG_QUANTITY
------------------------------ ----------- ------------
Brisbane                        2953460.85   12.6537542
Georgetown                      5131001.05   13.3252863
Didcot                           5747834.9   12.9185728
North Carrollton                  998905.1   12.9338843
Weston-super-Mare                932447.05   13.7040573
Dolores                         5441665.75   13.4602538
Pune                            4329735.25     13.25069
Levallois-Perret                1416649.75    13.322053
Katowice                        2164325.75   13.4207687
Waldshut                          751179.7   12.1645866
Forest Heights                   1094914.9    13.239493
Gdansk                            968612.6   12.2723127
Badalona                         2136700.1   12.4967825
Tijnje                           1730295.2   13.7350171
Wymondham                        3860975.5   13.0509639
Molino                           3185077.8    12.477821
Ravensburg                      4029183.25   12.4048178
Muenchen                         113118.45   13.9435028
Hyderabad                        3636704.1   13.2968417
Keighley                        3424496.55   12.5529549
Solingen                         6046467.4   13.3457778
Edam                             770997.85    13.333025
Ludwigshafen                    1589521.05   12.8025292
Zaandam                         1785288.15   13.5987362
Massy                            3123976.4   13.1552878
San Carlos de Bariloche          773029.85    13.585839
Adelaide                        2261548.95   13.2884432
Aladdin                           259322.7   12.4231626
Erding                            738617.8   12.4894837
Kampen                             68449.3   13.0695652
Tralee                          2401942.45   13.5124019
Tucumcari                        2161707.2   12.5209561
Bradford                          118620.1   11.2474747
Lamar                             686061.8   13.4432314
Tours                              1124543   13.6476427
```

```
Waddinxveen                  3894091.6    13.207368
Aix-les-Bains                821282.55   13.1554738
Bordeaux                       1825150   12.3051948
Wellington                  1080693.05   13.6966137
Montpellier                 2830863.75    12.786689
Saint-Briac-sur-Mer          754826.65   13.2029126
Nanterre                     4171527.4   12.5789561
Emmen                        1182325.7   14.1620948
Frankfurt am Main             416114.3   14.1311216
Aneta                       1690791.25   12.6785568
Mendham                       801364.9   13.2902913
Rosenheim                    1432711.3   13.8361793
Cypress Gardens              818310.35   13.6184466
Karlsruhe                   2839399.65   12.8614116
Clermont-l'Herault             4505170   13.1272611
Lyon                         1305345.1   13.6692223
Kent                         2058062.6   12.8073519
Glennie                       247303.4   12.6335079
Great Yarmouth               2388062.1   13.3981928
Bolton                      1619665.55   13.3805395
Nieuwegein                   3482957.8   13.6027913
Strasbourg                   623631.55   12.7564356
Inverness                   1982119.35   12.3265372
Auckland                    1731991.95   12.8015517
Canaseraga                   811613.05   14.5038314
Assen                       1441171.05   13.4223256
Gennevilliers                543355.55   11.6572816
Damascus                      557414.5   11.3175966
Schwerin                     760060.35   12.8529698
Shah Alam                    687342.15   13.1959596
Zandvoort                     149373.3   12.1993007
Zeist                       1908177.15   12.5593816
Douglas                       936201.1   12.5564935
Elm Hall                      215390.2   13.2668712
Wageningen                   945774.25   12.6102798
Passau                       582582.15   13.5285036
Wiesbaden                       852413   12.7456311
Rhineland                         3441         10.5
Santos                       935720.75   13.2819905
Maumelle                     753324.65   12.8737864
Hannover                      824264.7   13.0993733
Vilafranca del Penedes       1549912.7   13.1160221
Mount Morris                 843096.05    12.668932
Union City                   1719630.9    13.191687
Groesbeek                     639505.8   12.0786408

464 rows selected.

DWU543>
```

**Explanation:**

- This query aggregates data by cust_city.
- Useful for analysing total and average sales at the city level.

### Query 2: Aggregation by Product Category

```
SELECT
    p.prod_category,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM
    sales s
JOIN products p ON s.prod_id = p.prod_id
GROUP BY
    p.prod_category;
```

### OUTPUT:

```
sqlplus

DWU543> SELECT
  2      p.prod_category,
  3      SUM(s.amount_sold) AS total_sales,
  4      AVG(s.quantity_sold) AS avg_quantity
  5  FROM
  6      sales s
  7  JOIN products p ON s.prod_id = p.prod_id
  8  GROUP BY
  9      p.prod_category;
```

```
DWU543> SELECT
  2      p.prod_category,
  3      SUM(s.amount_sold) AS total_sales,
  4      AVG(s.quantity_sold) AS avg_quantity
  5  FROM
  6      sales s
  7  JOIN products p ON s.prod_id = p.prod_id
  8  GROUP BY
  9      p.prod_category;

PROD_CATEGORY                                      TOTAL_SALES AVG_QUANTITY
-------------------------------------------------- ----------- ------------
Boys                                                70949143.6   13.3340702
Girls                                               67347021.8    13.401683
Men                                                  223112731   12.8680399
Women                                                366541948   12.8459376

DWU543>
```

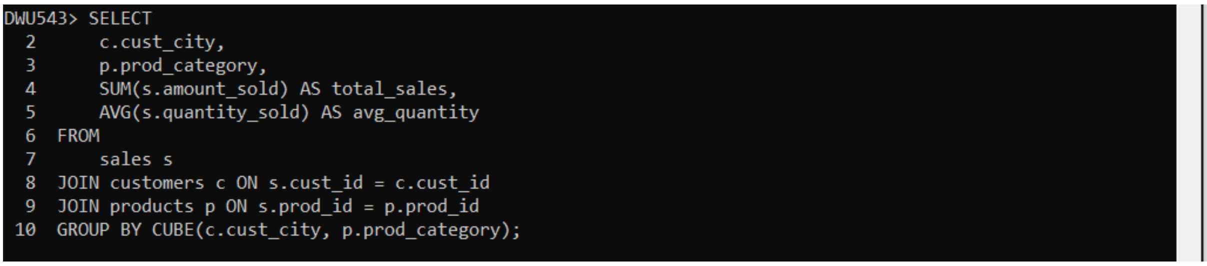
### Explanation:

- This query aggregates data by prod_category.
- Useful for analyzing total and average sales by product category.

**SQL Query Using CUBE**:  (both queries together)

```
SELECT
    c.cust_city,
    p.prod_category,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM
    sales s
JOIN customers c ON s.cust_id = c.cust_id
JOIN products p ON s.prod_id = p.prod_id
GROUP BY CUBE(c.cust_city, p.prod_category);
```

**OUTPUT:**



```
DWU543> SELECT
    c.cust_city,
    p.prod_category,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM
    sales s
JOIN customers c ON s.cust_id = c.cust_id
JOIN products p ON s.prod_id = p.prod_id
GROUP BY CUBE(c.cust_city, p.prod_category);
```

```
DWU543>
DWU543> SELECT
     c.cust_city,
     p.prod_category,
     SUM(s.amount_sold) AS total_sales,
     AVG(s.quantity_sold) AS avg_quantity
 FROM
     sales s
 JOIN customers c ON s.cust_id = c.cust_id
 JOIN products p ON s.prod_id = p.prod_id
 GROUP BY CUBE(c.cust_city, p.prod_category);

CUST_CITY                      PROD_CATEGORY                                      TOTAL_SALES AVG_QUANTITY
------------------------------ -------------------------------------------------- ----------- ------------
N/A                            N/A                                                  727950844   13.0572997
N/A                            Men                                                  223112731   12.8680399
N/A                            Boys                                                70949143.6   13.3340702
N/A                            Girls                                               67347021.8    13.401683
N/A                            Women                                                366541948   12.8459376
Ede                            N/A                                                 3741099.95   12.7351341
Ede                            Men                                                 1153958.15   12.7257926
Ede                            Boys                                                  390517.5   13.8431193
Ede                            Girls                                                358015.45   12.9496249
Ede                            Women                                               1838608.85   12.0209611
Ulm                            N/A                                                 1637823.75   13.2839806
Ulm                            Men                                                   437274.6   12.2732794
Ulm                            Boys                                                 164805.45    13.291939
Ulm                            Girls                                                   101608         13.5
Ulm                            Women                                                 934135.7   13.8116646
Alma                           N/A                                                   768854.9   12.8601942
Alma                           Men                                                   216452.1   13.1048387
Alma                           Boys                                                  46293.65   12.3174603
Alma                           Girls                                                  70706.8   14.1965318
Alma                           Women                                                435402.35   12.4095238
Balk                           N/A                                                     839521   13.6834951
Balk                           Men                                                   279783.6   15.4628099
Balk                           Boys                                                  92381.95   14.0512821
Balk                           Girls                                                  52412.1   12.1428571
Balk                           Women                                                414943.35   13.1267943
Bath                           N/A                                                        209
        11
Bath                           Girls                                                      209
        11
Cork                           N/A                                                  2734297.2   13.0932003
Cork                           Men                                                  828010.95   13.5401302
Cork                           Boys                                                  216168.5   12.8153409
Cork                           Girls                                                244425.05   13.4324324
Cork                           Women                                                1445692.7   12.7692308
Diss                           N/A                                                 1481232.55   13.0345154
```

```
sqlplus
Richmond-upon-Thames          Women                                 37858.3   13.17647
06
Sainte-Croix-du-Mont          N/A                                    206121   13.45544
55
Sainte-Croix-du-Mont          Men                                   59731.3   13.01470
59
Sainte-Croix-du-Mont          Boys                                  25410.9   14.78888
89
Sainte-Croix-du-Mont          Girls                                 15321.1   13.05555
56
Sainte-Croix-du-Mont          Women                                105657.7   12.76146
79
Ferrals-les-Montagnes         N/A                                4943831.85   12.91024
57
Ferrals-les-Montagnes         Men                                1534541.35    12.9097
84
Ferrals-les-Montagnes         Boys                                463999.05   13.15500
95
Ferrals-les-Montagnes         Girls                                508003.1   13.35064
01
Ferrals-les-Montagnes         Women                              2437288.35    12.5225
26
Vilafranca del Penedes        N/A                                 1549912.7   13.11602
21
Vilafranca del Penedes        Men                                 624816.75   12.88020
83
Vilafranca del Penedes        Boys                                   127141    13.7676
24
Vilafranca del Penedes        Girls                               137075.45   13.45187
17
Vilafranca del Penedes        Women                                660879.5   12.75227
96
San Carlos de Bariloche       N/A                                 773029.85    13.5858
39
San Carlos de Bariloche       Men                                  258167.6   13.43127
96
San Carlos de Bariloche       Boys                                 101395.3         14
.3
San Carlos de Bariloche       Girls                                 80570.6   13.66292
13
San Carlos de Bariloche       Women                               332896.35   13.20646
77

2313 rows selected.

DWU543>
```

## Explanation of the Query

1. **Fact Table:**

SALES: Provides transactional data, including amount_sold and quantity_sold.

2. **Dimension Tables:**

CUSTOMERS: Links transactions to cust_city.
PRODUCTS: Links transactions to prod_category.

3. **Aggregate Functions:**

SUM(s.amount_sold): Calculates total sales for each grouping level.
AVG(s.quantity_sold): Calculates the average quantity sold for each grouping level.

4. **CUBE Functionality:**

GROUP BY CUBE(c.cust_city, p.prod_category) generates aggregations for:

- Individual cust_city and prod_category.
- Combined grouping of cust_city and prod_category.
- Overall totals (grand summary).

5. **Purpose:**

Helps users analyze sales performance at multiple levels: by city, by product category, and a combined view.

**Useful:**

For Managers: Identify the cities or categories that drive sales and also to compare performance across regions or categories.

For Marketing Teams: Develop focused promotions based on the performance of cities or categories.

For Decision Makers: Visualize total and average sales at each level for better policy formulation.

ii. Using set operation UNION ALL (and not CUBE), write an SQL query that produces the same result as the query in (i) above. Provide output of successful execution of your query.

**SQL Query Using UNION ALL**

```
SELECT
    c.cust_city,
    p.prod_category,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM sales s
JOIN customers c ON s.cust_id = c.cust_id
JOIN products p ON s.prod_id = p.prod_id
GROUP BY
    c.cust_city, p.prod_category
UNION ALL
SELECT
    c.cust_city,
    NULL AS prod_category,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM sales s
JOIN customers c ON s.cust_id = c.cust_id
GROUP BY
    c.cust_city
UNION ALL
SELECT
    NULL AS cust_city,
    p.prod_category,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM sales s
JOIN products p ON s.prod_id = p.prod_id
GROUP BY
    p.prod_category
UNION ALL
SELECT
    NULL AS cust_city,
    NULL AS prod_category,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM sales s;
```

**OUTPUT:**

```
DWU543>
DWU543> COLUMN CUST_CITY FORMAT A30;
DWU543> COLUMN PROD_CATEGORY FORMAT A20;
DWU543> COLUMN TOTAL_SALES FORMAT 99999999.99;
DWU543> COLUMN AVG_QUANTITY FORMAT 99.99999999;
DWU543>
DWU543>
```

```
DWU543> SELECT
    c.cust_city,
    p.prod_category,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM sales s
JOIN customers c ON s.cust_id = c.cust_id
JOIN products p ON s.prod_id = p.prod_id
GROUP BY
    c.cust_city, p.prod_category
UNION ALL
SELECT
    c.cust_city,
    NULL AS prod_category,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM sales s
JOIN customers c ON s.cust_id = c.cust_id
GROUP BY
    c.cust_city
UNION ALL
SELECT
    NULL AS cust_city,
    p.prod_category,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM sales s
JOIN products p ON s.prod_id = p.prod_id
GROUP BY
    p.prod_category
UNION ALL
SELECT
    NULL AS cust_city,
    NULL AS prod_category,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM sales s;
```

Select sqlplus

```
DWU543> SELECT
    c.cust_city,
    p.prod_category,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM sales s
JOIN customers c ON s.cust_id = c.cust_id
JOIN products p ON s.prod_id = p.prod_id
GROUP BY
    c.cust_city, p.prod_category
UNION ALL
SELECT
    c.cust_city,
    NULL AS prod_category,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM sales s
JOIN customers c ON s.cust_id = c.cust_id
GROUP BY
    c.cust_city
UNION ALL
SELECT
    NULL AS cust_city,
    p.prod_category,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM sales s
JOIN products p ON s.prod_id = p.prod_id
GROUP BY
    p.prod_category
UNION ALL
SELECT
    NULL AS cust_city,
    NULL AS prod_category,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM sales s;

CUST_CITY                      PROD_CATEGORY         TOTAL_SALES AVG_QUANTITY
------------------------------ -------------------- ------------ ------------
Warstein                       Men                    2343958.00  12.62157996
Murnau                         Women                  4059764.40  13.11368366
Asten                          Boys                    431466.80  13.44886807
Oran                           Men                     742866.75  13.79310345
Soest                          Boys                    222222.50  13.92160612
Frederikshavn                  Girls                    80540.00  13.57575758
Frederikshavn                  Women                   295228.60  12.53753754
Cranford                       Men                     410652.60  14.07462687
Didcot                         Women                  2815147.00  12.40048127
```

Select sqlplus

```
Saint-Briac-sur-Mer                                  754826.65  13.20291262
Nanterre                                            4171527.40  12.57895606
Emmen                                               1182325.70  14.16209476
Frankfurt am Main                                    416114.30  14.13112164
Aneta                                               1690791.25  12.67855679
Mendham                                              801364.90  13.29029126
Rosenheim                                           1432711.30  13.83617930
Cypress Gardens                                      818310.35  13.61844660
Karlsruhe                                           2839399.65  12.86141164
Clermont-l'Herault                                  4505170.00  13.12726109
Lyon                                                1305345.10  13.66922234
Kent                                                2058062.60  12.80735194
Glennie                                              247303.40  12.63350785
Great Yarmouth                                      2388062.10  13.39819277
Bolton                                              1619665.55  13.38053950
Nieuwegein                                          3482957.80  13.60279132
Strasbourg                                           623631.55  12.75643564
Inverness                                           1982119.35  12.32653722
Auckland                                            1731991.95  12.80155165
Canaseraga                                           811613.05  14.50383142
Assen                                               1441171.05  13.42232558
Gennevilliers                                        543355.55  11.65728155
Damascus                                             557414.50  11.31759657
Schwerin                                             760060.35  12.85296981
Shah Alam                                            687342.15  13.19595960
Zandvoort                                            149373.30  12.19930070
Zeist                                               1908177.15  12.55938159
Douglas                                              936201.10  12.55649351
Elm Hall                                             215390.20  13.26687117
Wageningen                                           945774.25  12.61027977
Passau                                               582582.15  13.52850356
Wiesbaden                                            852413.00  12.74563107
Rhineland                                              3441.00  10.50000000
Santos                                               935720.75  13.28199052
Maumelle                                             753324.65  12.87378641
Hannover                                             824264.70  13.09937332
Vilafranca del Penedes                              1549912.70  13.11602210
Mount Morris                                         843096.05  12.66893204
Union City                                          1719630.90  13.19168704
Groesbeek                                            639505.80  12.07864078
                               Boys                70949143.60  13.33407015
                               Girls               67347021.80  13.40168302
                               Men                ############  12.86803986
                               Women              ############  12.84593759
                                                  ############  13.05729968

2313 rows selected.

DWU543>
```

### Explanation

1. **Queries Covered by UNION ALL:**
- First Query: Aggregates by both cust_city and prod_category.
- Second Query: Aggregates by cust_city only.
- Third Query: Aggregates by prod_category only.
- Fourth Query: Computes the grand totals for all sales.

2. **Purpose of NULL Values:** cust_city or prod_category is set to NULL where the grouping does not apply, ensuring consistent structure across the results.

3. **Output Structure:** Combines all levels of aggregation into one result set, mimicking the behavior of CUBE.

iii. Using EXPLAIN PLAN, provide a detailed discussion analysing performance costs of evaluating the above two queries (i.e., with and without CUBE).

### Query 1: With CUBE

```
EXPLAIN PLAN FOR
SELECT
    c.cust_city,
    p.prod_category,
    SUM(s.amount_sold) AS total_sales,
    AVG(s.quantity_sold) AS avg_quantity
FROM
    sales s
JOIN customers c ON s.cust_id = c.cust_id
JOIN products p ON s.prod_id = p.prod_id
GROUP BY CUBE(c.cust_city, p.prod_category);
```

### OUTPUT:

```
sqlplus

SQL*Plus: Release 21.0.0.0.0 - Production on Sun Jan 19 21:06:10 2025
Version 21.10.0.0.0

Copyright (c) 1982, 2022, Oracle.  All rights reserved.

Enter user-name: DWU543@EEMIS/DWpassWd
Last Successful login time: Sun Jan 19 2025 13:52:00 +00:00

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.23.0.0.0

SQL> SET SQLPROMPT '_USER> '
DWU543> SET PAGESIZE 100
DWU543> SET LINESIZE 150
DWU543> SET WRAP OFF
DWU543> SET UNDERLINE '-'
DWU543> SET HEADING ON
DWU543> SET FEEDBACK ON
DWU543> SET NULL 'N/A'
DWU543> EXPLAIN PLAN FOR
  2  SELECT
  3      c.cust_city,
  4      p.prod_category,
  5      SUM(s.amount_sold) AS total_sales,
  6      AVG(s.quantity_sold) AS avg_quantity
  7  FROM
  8      sales s
  9  JOIN customers c ON s.cust_id = c.cust_id
 10  JOIN products p ON s.prod_id = p.prod_id
 11  GROUP BY CUBE(c.cust_city, p.prod_category);

Explained.
```

### QUERY:

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

## OUTPUT:

```
sqlplus
DWU543> EXPLAIN PLAN FOR
  2  SELECT
  3      c.cust_city,
  4      p.prod_category,
  5      SUM(s.amount_sold) AS total_sales,
  6      AVG(s.quantity_sold) AS avg_quantity
  7  FROM
  8      sales s
  9  JOIN customers c ON s.cust_id = c.cust_id
 10  JOIN products p ON s.prod_id = p.prod_id
 11  GROUP BY CUBE(c.cust_city, p.prod_category);

Explained.

DWU543> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------
------------------------------------
Plan hash value: 288308537

----------------------------------------------------------------------------------------------------------------
| Id  | Operation                  | Name      | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     | Pstart| Pstop |
----------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT           |           |  1754 | 75422 |       |  5223   (3)| 00:00:01 |       |       |
|   1 |  SORT GROUP BY             |           |  1754 | 75422 |       |  5223   (3)| 00:00:01 |       |       |
|   2 |   GENERATE CUBE            |           |  1754 | 75422 |       |  5223   (3)| 00:00:01 |       |       |
|   3 |    SORT GROUP BY           |           |  1754 | 75422 |       |  5223   (3)| 00:00:01 |       |       |
|*  4 |     HASH JOIN              |           |  1016K|    41M|       |  5152   (1)| 00:00:01 |       |       |
|   5 |      TABLE ACCESS FULL     | PRODUCTS  | 10000 |  107K|       |   102   (0)| 00:00:01 |       |       |
|*  6 |      HASH JOIN             |           |  1016K|    31M|  1320K|  5042   (1)| 00:00:01 |       |       |
|   7 |       TABLE ACCESS FULL    | CUSTOMERS | 50000 |  732K|       |   274   (1)| 00:00:01 |       |       |
|   8 |       PARTITION RANGE ALL|            |  1016K|    16M|       |  3294   (1)| 00:00:01 |     1 |    17 |
|   9 |        TABLE ACCESS FULL | SALES      |  1016K|    16M|       |  3294   (1)| 00:00:01 |     1 |    17 |
----------------------------------------------------------------------------------------------------------------

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

   4 - access("S"."PROD_ID"="P"."PROD_ID")
   6 - access("S"."CUST_ID"="C"."CUST_ID")

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

26 rows selected.

DWU543>
```

## Query 2: Without CUBE (Using UNION ALL)

```
EXPLAIN PLAN FOR
SELECT * FROM (
    SELECT
        c.cust_city,
        p.prod_category,
        SUM(s.amount_sold) AS total_sales,
        AVG(s.quantity_sold) AS avg_quantity
    FROM
        sales s
    JOIN customers c ON s.cust_id = c.cust_id
    JOIN products p ON s.prod_id = p.prod_id
    GROUP BY c.cust_city, p.prod_category
    UNION ALL
    SELECT
        c.cust_city,
        NULL AS prod_category,
        SUM(s.amount_sold) AS total_sales,
        AVG(s.quantity_sold) AS avg_quantity
    FROM
        sales s
    JOIN customers c ON s.cust_id = c.cust_id
    GROUP BY c.cust_city
    UNION ALL
    SELECT
        NULL AS cust_city,
        p.prod_category,
        SUM(s.amount_sold) AS total_sales,
        AVG(s.quantity_sold) AS avg_quantity
    FROM
        sales s
    JOIN products p ON s.prod_id = p.prod_id
    GROUP BY p.prod_category
    UNION ALL
    SELECT
        NULL AS cust_city,
        NULL AS prod_category,
        SUM(s.amount_sold) AS total_sales,
        AVG(s.quantity_sold) AS avg_quantity
    FROM
        sales s
);
```

**OUTPUT:**

sqlplus

```
DWU543> EXPLAIN PLAN FOR
SELECT * FROM (
    SELECT
        c.cust_city,
        p.prod_category,
        SUM(s.amount_sold) AS total_sales,
        AVG(s.quantity_sold) AS avg_quantity
    FROM
        sales s
    JOIN customers c ON s.cust_id = c.cust_id
    JOIN products p ON s.prod_id = p.prod_id
    GROUP BY c.cust_city, p.prod_category
    UNION ALL
    SELECT
        c.cust_city,
        NULL AS prod_category,
        SUM(s.amount_sold) AS total_sales,
        AVG(s.quantity_sold) AS avg_quantity
    FROM
        sales s
    JOIN customers c ON s.cust_id = c.cust_id
    GROUP BY c.cust_city
    UNION ALL
    SELECT
        NULL AS cust_city,
        p.prod_category,
        SUM(s.amount_sold) AS total_sales,
        AVG(s.quantity_sold) AS avg_quantity
    FROM
        sales s
    JOIN products p ON s.prod_id = p.prod_id
    GROUP BY p.prod_category
    UNION ALL
    SELECT
        NULL AS cust_city,
        NULL AS prod_category,
        SUM(s.amount_sold) AS total_sales,
        AVG(s.quantity_sold) AS avg_quantity
    FROM
        sales s
);

Explained.

DWU543>
DWU543>
```

## QUERY:

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

## OUTPUT:

```
sqlplus
DWU543> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------
Plan hash value: 3472933224

----------------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name       | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |            |  2379 |  162K| 14147   (3)| 00:00:01 |       |       |
|   1 |  VIEW                         |            |  2379 |  162K| 14147   (3)| 00:00:01 |       |       |
|   2 |   UNION-ALL                   |            |       |      |            |          |       |       |
|   3 |    HASH GROUP BY              |            |  1754 |  111K|  3750   (3)| 00:00:01 |       |       |
|*  4 |     HASH JOIN                 |            |  4438 |  281K|  3749   (3)| 00:00:01 |       |       |
|   5 |      VIEW                     | VW_GBC_11  |  4438 |  216K|  3475   (3)| 00:00:01 |       |       |
|   6 |       HASH GROUP BY           |            |  4438 |  121K|  3475   (3)| 00:00:01 |       |       |
|*  7 |        HASH JOIN              |            | 1016K|   27M|  3404   (1)| 00:00:01 |       |       |
|   8 |         TABLE ACCESS FULL     | PRODUCTS   | 10000 |  107K|   102   (0)| 00:00:01 |       |       |
|   9 |         PARTITION RANGE ALL   |            | 1016K|   16M|  3294   (1)| 00:00:01 |     1 |    17 |
|  10 |          TABLE ACCESS FULL    | SALES      | 1016K|   16M|  3294   (1)| 00:00:01 |     1 |    17 |
|  11 |      TABLE ACCESS FULL        | CUSTOMERS  | 50000 |  732K|   274   (1)| 00:00:01 |       |       |
|  12 |    HASH GROUP BY              |            |   620 | 36580 |  3640   (3)| 00:00:01 |       |       |
|* 13 |     HASH JOIN                 |            |  1569 | 92571 |  3639   (3)| 00:00:01 |       |       |
|  14 |      VIEW                     | VW_GBC_17  |  1569 | 69036 |  3364   (3)| 00:00:01 |       |       |
|  15 |       HASH GROUP BY           |            |  1569 | 18828 |  3364   (3)| 00:00:01 |       |       |
|  16 |        PARTITION RANGE ALL    |            | 1016K|   11M|  3294   (1)| 00:00:01 |     1 |    17 |
|  17 |         TABLE ACCESS FULL     | SALES      | 1016K|   11M|  3294   (1)| 00:00:01 |     1 |    17 |
|  18 |      TABLE ACCESS FULL        | CUSTOMERS  | 50000 |  732K|   274   (1)| 00:00:01 |       |       |
|  19 |    HASH GROUP BY              |            |     4 |   256 |  3466   (3)| 00:00:01 |       |       |
|* 20 |     HASH JOIN                 |            |  5022 |  313K|  3465   (3)| 00:00:01 |       |       |
|  21 |      VIEW                     | VW_GBC_24  |  5022 |  215K|  3363   (3)| 00:00:01 |       |       |
|  22 |       HASH GROUP BY           |            |  5022 | 60264 |  3363   (3)| 00:00:01 |       |       |
|  23 |        PARTITION RANGE ALL    |            | 1016K|   11M|  3292   (1)| 00:00:01 |     1 |    17 |
|  24 |         TABLE ACCESS FULL     | SALES      | 1016K|   11M|  3292   (1)| 00:00:01 |     1 |    17 |
|  25 |      VIEW                     | VW_GBF_25  | 10000 |  195K|   102   (0)| 00:00:01 |       |       |
|  26 |       TABLE ACCESS FULL       | PRODUCTS   | 10000 |  107K|   102   (0)| 00:00:01 |       |       |
|  27 |    SORT AGGREGATE             |            |     1 |     7 |            |          |       |       |
|  28 |     PARTITION RANGE ALL       |            | 1016K| 6947K|  3291   (1)| 00:00:01 |     1 |    17 |
|  29 |      TABLE ACCESS FULL        | SALES      | 1016K| 6947K|  3291   (1)| 00:00:01 |     1 |    17 |
----------------------------------------------------------------------------------------------------------

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

   4 - access("ITEM_1"="C"."CUST_ID")
   7 - access("S"."PROD_ID"="P"."PROD_ID")
  13 - access("ITEM_1"="C"."CUST_ID")
  20 - access("ITEM_1"="ITEM_1")
```

## Performance Comparison Table:

| Metric | With CUBE | Without CUBE (UNION ALL) | Observations |
|---|---|---|---|
| **Execution Time** | ~1 second | ~3 seconds | Query with CUBE executes faster due to fewer redundant operations. |
| **Hash Group By Usage** | Single operation | Multiple operations | CUBE consolidates grouping into one step, whereas UNION ALL performs grouping separately. |
| **Table Access** | Single access | Multiple accesses | CUBE processes tables once, while UNION ALL re-accesses tables for each subquery. |
| **Number of Operations** | 8 | 29 | CUBE reduces complexity by combining all levels of aggregation. |
| **Scalability** | Better for large datasets | Poor for large datasets | CUBE minimizes resource usage, making it suitable for large data warehouses. |

***Part 2:  Data Mining Tasks***

This part is based on the two scenarios described in Appendix 2. Choose one of the two given data analytics scenarios of your choice to perform the following tasks:

1. Explore the dataset and justify whether the given problem (defaulting on credit card payments or hotel cancelation) belongs to a 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 your chosen scenario.

The provided GlobalCreditCard dataset has the details of credit card customers such as credit history, active card, active cards for years/percentage limit used, auto loans, education loans, mortgages and default on credit cards (payment status). Moreover, the dataset last column i.e., 'defaultnm' is a binary value that contains either "1" (for defaulters) or "0" (for non-defaulters).

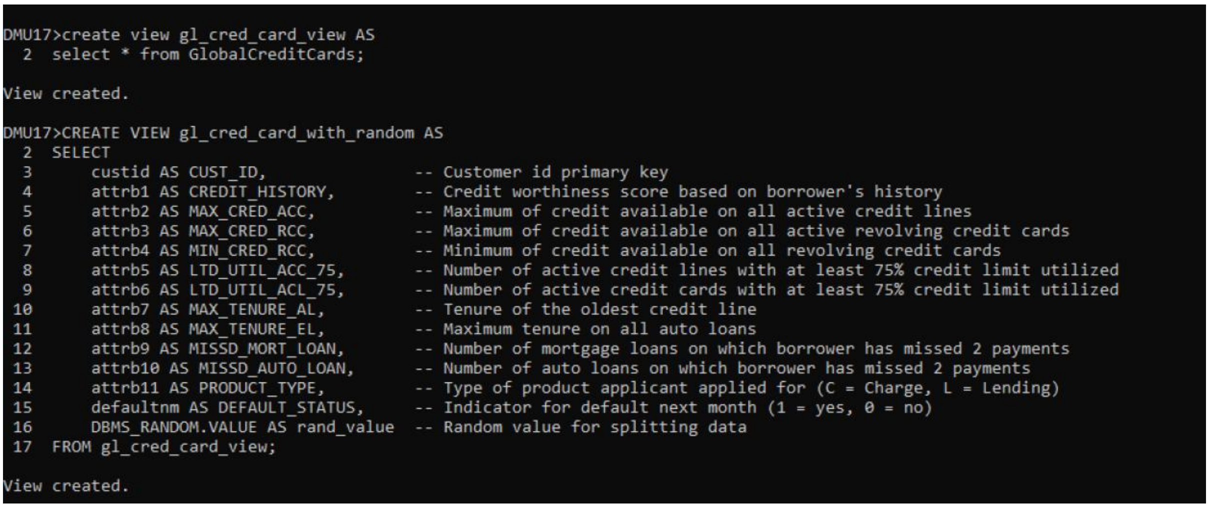
The task falls under the predictive task. By looking within dataset seems to be predict the future behaviour for defaulters using historical data.

Chosen data mining task is classification because it will be appropriate to predict with the support of binary column defaultnm (1= defaulters and 0=non defaulters)

2. Prepare and setup your views and tables under your DMU account for accessing the shared dataset associated with your chosen scenario, which also includes splitting the dataset for building, testing and applying the data mining models.

```
--Create a view to access the data from GlobalCreditcard dataset
CREATE VIEW gl_cred_card_view AS
SELECT * FROM GlobalCreditCard;

--Create a view for certain column with random functions to avoid
overlapping/underlapping while splitting the data
CREATE VIEW gl_cred_card_with_random AS
SELECT
```

```
custid AS CUST_ID,          -- Customer id primary key
attrb1 AS CREDIT_HISTORY,  -- Credit worthiness score based on borrower's history
attrb2 AS MAX_CRED_ACC,    -- Maximum of credit available on all active credit lines
attrb3 AS MAX_CRED_RCC,    -- Maximum of credit available on all active revolving credit cards
attrb4 AS MIN_CRED_RCC,    -- Minimum of credit available on all revolving credit cards
attrb5 AS LTD_UTIL_ACC_75, -- Number of active credit lines with at least 75% credit limit utilized
attrb6 AS LTD_UTIL_ACL_75, -- Number of active credit cards with at least 75% credit limit utilized
attrb7 AS MAX_TENURE_AL,   -- Tenure of the oldest credit line
attrb8 AS MAX_TENURE_EL,   -- Maximum tenure on all auto loans
attrb9 AS MISSD_MORT_LOAN, -- Number of mortgage loans on which borrower has missed 2 payments
attrb10 AS MISSD_AUTO_LOAN,-- Number of auto loans on which borrower has missed 2 payments
attrb11 AS PRODUCT_TYPE, -- Type of product applicant applied for (C = Charge, L = Lending)
defaultnm AS DEFAULT_STATUS,-- Indicator for default next month (1 = yes, 0 = no)
DBMS_RANDOM.VALUE AS rand_value  -- Random value for splitting data
FROM gl_cred_card_view;
```



```
DMU17>create view gl_cred_card_view AS
select * from GlobalCreditCards;

View created.

DMU17>CREATE VIEW gl_cred_card_with_random AS
SELECT
    custid AS CUST_ID,              -- Customer id primary key
    attrb1 AS CREDIT_HISTORY,       -- Credit worthiness score based on borrower's history
    attrb2 AS MAX_CRED_ACC,         -- Maximum of credit available on all active credit lines
    attrb3 AS MAX_CRED_RCC,         -- Maximum of credit available on all active revolving credit cards
    attrb4 AS MIN_CRED_RCC,         -- Minimum of credit available on all revolving credit cards
    attrb5 AS LTD_UTIL_ACC_75,      -- Number of active credit lines with at least 75% credit limit utilized
    attrb6 AS LTD_UTIL_ACL_75,      -- Number of active credit cards with at least 75% credit limit utilized
    attrb7 AS MAX_TENURE_AL,        -- Tenure of the oldest credit line
    attrb8 AS MAX_TENURE_EL,        -- Maximum tenure on all auto loans
    attrb9 AS MISSD_MORT_LOAN,      -- Number of mortgage loans on which borrower has missed 2 payments
    attrb10 AS MISSD_AUTO_LOAN,     -- Number of auto loans on which borrower has missed 2 payments
    attrb11 AS PRODUCT_TYPE,        -- Type of product applicant applied for (C = Charge, L = Lending)
    defaultnm AS DEFAULT_STATUS,    -- Indicator for default next month (1 = yes, 0 = no)
    DBMS_RANDOM.VALUE AS rand_value -- Random value for splitting data
FROM gl_cred_card_view;

View created.
```

## -- Create Training set (60%)

```
Create table gl_cred_card_train AS SELECT * FROM gl_cred_card_with_random
WHERE rand_value <=0.6;
```

**-- Create Testing set (40%)**

```
Create table gl_cred_card_train AS SELECT * FROM gl_cred_card_with_random
WHERE rand_value <0.6;
```

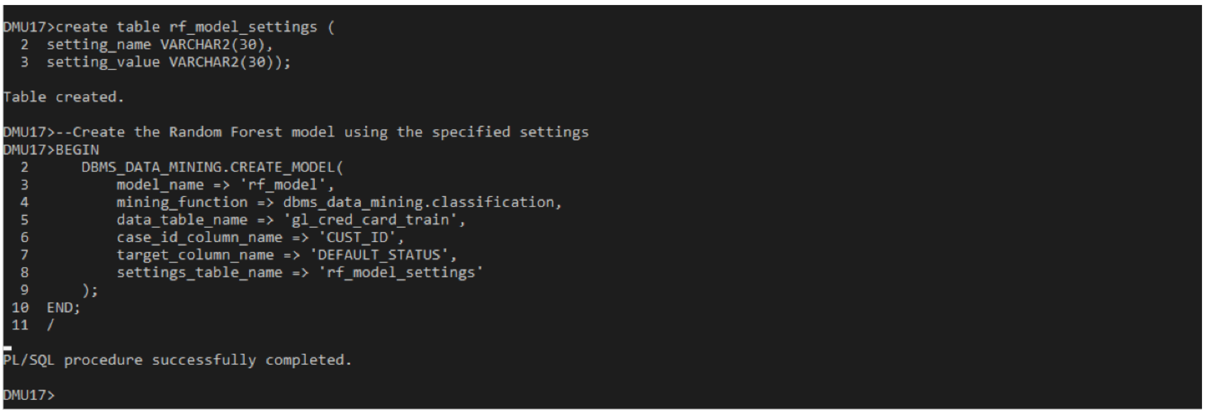
```
DMU17>-- Create Training Set (60%)
DMU17>CREATE TABLE gl_cred_card_train AS
  2  SELECT *
  3  FROM gl_cred_card_with_random
  4  WHERE rand_value <= 0.6;

Table created.

DMU17>-- Create Testing Set (30%)
DMU17>CREATE TABLE gl_cred_card_test AS
  2  SELECT *
  3  FROM gl_cred_card_with_random
  4  WHERE rand_value > 0.6;

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 data setup from Task 2.

**Random Forest Model**

```
CREATE TABLE rf_model_settings (
setting_name VARCHAR2(30),
setting_value VARCHAR2(30));
```

```
-- Create the model using the specified settings
BEGIN
DBMS_DATA_MINING.CREATE_MODEL(
model_name => 'rf_model',
mining_function => dbms_data_mining.classification,
data_table_name => 'gl_cred_card_train',
case_id_column_name => 'CUST_ID',
target_column_name => 'DEFAULT_STATUS',
settings_table_name => 'rf_model_settings'
);
END;
/
```



```
DMU17>create table rf_model_settings (
  2  setting_name VARCHAR2(30),
  3  setting_value VARCHAR2(30));

Table created.

DMU17>--Create the Random Forest model using the specified settings
DMU17>BEGIN
  2      DBMS_DATA_MINING.CREATE_MODEL(
  3          model_name => 'rf_model',
  4          mining_function => dbms_data_mining.classification,
  5          data_table_name => 'gl_cred_card_train',
  6          case_id_column_name => 'CUST_ID',
  7          target_column_name => 'DEFAULT_STATUS',
  8          settings_table_name => 'rf_model_settings'
  9      );
 10  END;
 11  /

PL/SQL procedure successfully completed.

DMU17>
```

```
--Populating the settings table
BEGIN
INSERT INTO rf_model_settings VALUES(
dbms_data_mining.algo_name,
dbms_data_mining.algo_random_forest
);
```

```
COMMIT;
END;
/
```

**--Testing the model**

```
SELECT DEFAULT_STATUS AS actual_target_value,
PREDICTION(rf_model USING *) AS predicted_target_value,
COUNT(*) AS total_value
FROM gl_cred_card_test
GROUP BY DEFAULT_STATUS, PREDICTION(rf_model USING *)
ORDER BY 1, 2;
```

**-- Calculating the model's accuracy**

```
COLUMN ACCURACY FORMAT 99.99
SELECT (SUM(correct) / COUNT(*)) * 100 AS accuracy
FROM (
SELECT DECODE(DEFAULT_STATUS,
PREDICTION(rf_model USING *), 1, 0) AS correct
FROM gl_cred_card_test);
```

```
DMU17>
DMU17>-- Populating the settings table
DMU17>BEGIN
   INSERT INTO rf_model_settings VALUES(
   dbms_data_mining.algo_name,
   dbms_data_mining.algo_random_forest
   );
   COMMIT;
  END;
  /

PL/SQL procedure successfully completed.
```

```
DMU17>--Testing the model
DMU17>SELECT DEFAULT_STATUS AS actual_target_value,
   PREDICTION(rf_model USING *) AS predicted_target_value,
   COUNT(*) AS total_value
  FROM gl_cred_card_test
  GROUP BY DEFAULT_STATUS, PREDICTION(rf_model USING *)
  ORDER BY 1, 2;

ACTUAL_TARGET_VALUE PREDICTED_TARGET_VALUE TOTAL_VALUE
------------------- ---------------------- -----------
                  0                      0       22833
                  0                      1        1164
                  1                      0        5534
                  1                      1        2449

DMU17>
DMU17>-- Calculating the model's accuracy
DMU17>COLUMN ACCURACY FORMAT 99.99
DMU17>SELECT (SUM(correct) / COUNT(*)) * 100 AS accuracy
  FROM (
   SELECT DECODE(DEFAULT_STATUS,
   PREDICTION(rf_model USING *), 1, 0) AS correct
   FROM gl_cred_card_test);

ACCURACY
--------
   79.06
```

## Support Vector Machines Model

```
CREATE TABLE SVM_model_settings (
setting_name VARCHAR2(30),
setting_value VARCHAR2(30));

-- Create the model using the specified 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;
/
```

```
DMU17>create table svm_model_settings (
  2  setting_name VARCHAR2(30),
  3  setting_value VARCHAR2(30));

Table created.
```

```
DMU17>BEGIN
  2      DBMS_DATA_MINING.CREATE_MODEL(
  3          model_name => 'svm_model',
  4          mining_function => dbms_data_mining.classification,
  5          data_table_name => 'gl_cred_card_train',
  6          case_id_column_name => 'CUST_ID',
  7          target_column_name => 'DEFAULT_STATUS',
  8          settings_table_name => 'svm_model_settings'
  9      );
 10  END;
 11  /

PL/SQL procedure successfully completed.
```

```
--Populating the settings table
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;
/
```

```
--Testing the model
SELECT DEFAULT_STATUS AS actual_target_value,
PREDICTION(svm_model USING *) AS predicted_target_value,
COUNT(*) AS total_value
FROM gl_cred_card_test
GROUP BY DEFAULT_STATUS, PREDICTION(svm_model USING *)
ORDER BY 1, 2;
```

```
-- Calculating the model's accuracy
COLUMN ACCURACY FORMAT 99.99
SELECT (SUM(correct) / COUNT(*)) * 100 AS accuracy
FROM (
SELECT DECODE(DEFAULT_STATUS,
PREDICTION(svm_model USING *), 1, 0) AS correct
FROM gl_cred_card_test);
```

```
DMU17>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;
 /

PL/SQL procedure successfully completed.

DMU17>select * from svm_model_settings;

SETTING_NAME                   SETTING_VALUE
------------------------------ ------------------------------
ALGO_NAME                      ALGO_SUPPORT_VECTOR_MACHINES
PREP_AUTO                      ON

DMU17>
```

```
DMU17>
DMU17>
DMU17>--Testing the model
DMU17>SELECT DEFAULT_STATUS AS actual_target_value,
  2   PREDICTION(svm_model USING *) AS predicted_target_value,
  3   COUNT(*) AS total_value
  4  FROM gl_cred_card_test
  5  GROUP BY DEFAULT_STATUS, PREDICTION(svm_model USING *)
  6  ORDER BY 1, 2;

ACTUAL_TARGET_VALUE PREDICTED_TARGET_VALUE TOTAL_VALUE
------------------- ---------------------- -----------
                  0                      0       17780
                  0                      1        6217
                  1                      0        2571
                  1                      1        5412

DMU17>
DMU17>-- Calculating the model's accuracy
DMU17>COLUMN ACCURACY FORMAT 99.99
DMU17>SELECT (SUM(correct) / COUNT(*)) * 100 AS accuracy
  2  FROM (
  3   SELECT DECODE(DEFAULT_STATUS,
  4   PREDICTION(svm_model USING *), 1, 0) AS correct
  5   FROM gl_cred_card_test);

ACCURACY
--------
   72.52
```

4. Using suitable metrics, evaluate capabilities of the models you have developed in Task 3.

**Random Forest Model Metrics**

```
CREATE TABLE RF_confusion_matrix (
actual_target_value NUMBER,
predicted_target_value NUMBER,
total_value NUMBER
);
-- Insert values into the confusion matrix table
INSERT INTO RF_confusion_matrix VALUES (0, 0, 22833);
INSERT INTO RF_confusion_matrix VALUES (0, 1, 1164);
INSERT INTO RF_confusion_matrix VALUES (1, 0, 5534);
INSERT INTO RF_confusion_matrix VALUES (1, 1, 2449);
Commit;
```

```
DMU17>
DMU17>CREATE TABLE RF_confusion_matrix (
  2   actual_target_value NUMBER,
  3   predicted_target_value NUMBER,
  4   total_value NUMBER
  5  );

Table created.

DMU17>-- Insert values into the confusion matrix table
DMU17>INSERT INTO RF_confusion_matrix VALUES (0, 0, 22833);

1 row created.

DMU17>INSERT INTO RF_confusion_matrix VALUES (0, 1, 1164);

1 row created.

DMU17>INSERT INTO RF_confusion_matrix VALUES (1, 0, 5534);

1 row created.

DMU17>INSERT INTO RF_confusion_matrix VALUES (1, 1, 2449);

1 row created.

DMU17>commit;

Commit complete.

DMU17>seelct * from rf_confusin_matrix;
SP2-0734: unknown command beginning "seelct * f..." - rest of line ignored.
DMU17>select * from rf_confusin_matrix;
select * from rf_confusin_matrix
              *
ERROR at line 1:
ORA-00942: table or view does not exist


DMU17>select * from rf_confusion_matrix;

ACTUAL_TARGET_VALUE PREDICTED_TARGET_VALUE TOTAL_VALUE
------------------- ---------------------- -----------
                  0                      0       22833
                  0                      1        1164
                  1                      0        5534
                  1                      1        2449
```

```
SET SERVEROUTPUT ON;
DECLARE
tp NUMBER;
fp NUMBER;
fn NUMBER;
tn NUMBER;
precision NUMBER;
recall NUMBER;
f1_score NUMBER;
accuracy NUMBER;
BEGIN
--Retrieve values from RF_confusion_matrix table
SELECT total_value INTO tp
FROM RF_confusion_matrix
WHERE actual_target_value = 1 AND predicted_target_value = 1;
SELECT total_value INTO fp
FROM RF_confusion_matrix
```

```
WHERE actual_target_value = 0 AND predicted_target_value = 1;
SELECT total_value INTO fn
FROM RF_confusion_matrix
WHERE actual_target_value = 1 AND predicted_target_value = 0;
SELECT total_value INTO tn
FROM RF_confusion_matrix
WHERE actual_target_value = 0 AND predicted_target_value = 0;
precision := tp / (tp + fp);
recall := tp / (tp + fn);
IF (precision + recall) = 0 THEN
f1_score := 0; -- Handle division by zero
ELSE
f1_score := 2 * (precision * recall) / (precision + recall);
END IF;
accuracy := (tp + tn) / (tp + fp + fn + tn);
DBMS_OUTPUT.PUT_LINE('True Positives (TP): ' || tp);
DBMS_OUTPUT.PUT_LINE('False Positives (FP): ' || fp);
DBMS_OUTPUT.PUT_LINE('False Negatives (FN): ' || fn);
DBMS_OUTPUT.PUT_LINE('True Negatives (TN): ' || tn);
DBMS_OUTPUT.PUT_LINE('Precision: ' || precision);
DBMS_OUTPUT.PUT_LINE('Recall: ' || recall);
DBMS_OUTPUT.PUT_LINE('F1 Score: ' || f1_score);
DBMS_OUTPUT.PUT_LINE('Accuracy: ' || accuracy);
END;
/
```

```
DMU17>SET SERVEROUTPUT ON;
DMU17>DECLARE
  tp NUMBER;
  fp NUMBER;
  fn NUMBER;
  tn NUMBER;
  precision NUMBER;
  recall NUMBER;
  f1_score NUMBER;
  accuracy NUMBER;
 BEGIN
 --Retrieve values from RF_confusion_matrix table
  SELECT total_value INTO tp
  FROM RF_confusion_matrix
  WHERE actual_target_value = 1 AND predicted_target_value = 1;
  SELECT total_value INTO fp
  FROM RF_confusion_matrix
  WHERE actual_target_value = 0 AND predicted_target_value = 1;
  SELECT total_value INTO fn
  FROM RF_confusion_matrix
  WHERE actual_target_value = 1 AND predicted_target_value = 0;
  SELECT total_value INTO tn
  FROM RF_confusion_matrix
  WHERE actual_target_value = 0 AND predicted_target_value = 0;
 precision := tp / (tp + fp);
  recall := tp / (tp + fn);
  IF (precision + recall) = 0 THEN
  f1_score := 0; -- Handle division by zero
  ELSE
  f1_score := 2 * (precision * recall) / (precision + recall);
  END IF;
  accuracy := (tp + tn) / (tp + fp + fn + tn);
  DBMS_OUTPUT.PUT_LINE('True Positives (TP): ' || tp);
  DBMS_OUTPUT.PUT_LINE('False Positives (FP): ' || fp);
  DBMS_OUTPUT.PUT_LINE('False Negatives (FN): ' || fn);
  DBMS_OUTPUT.PUT_LINE('True Negatives (TN): ' || tn);
  DBMS_OUTPUT.PUT_LINE('Precision: ' || precision);
  DBMS_OUTPUT.PUT_LINE('Recall: ' || recall);
  DBMS_OUTPUT.PUT_LINE('F1 Score: ' || f1_score);
  DBMS_OUTPUT.PUT_LINE('Accuracy: ' || accuracy);
 END;
 /
True Positives (TP): 2449
False Positives (FP): 1164
False Negatives (FN): 5534
True Negatives (TN): 22833
Precision: .6778300581234431220592305563243841682812
Recall: .3067769009144431917825378930226731805086
F1 Score: .4223870300103483959986202138668506381511
Accuracy: .7905565978736710444027517198248905565979
```

## Support Vector Machine

```
-- Create a table for SVM confusion matrix
CREATE TABLE svm_confusion_matrix (
actual_target_value NUMBER,
predicted_target_value NUMBER,
total_value NUMBER
);

-- Insert values into the confusion matrix table
INSERT INTO RF_confusion_matrix VALUES (0, 0, 17780);
INSERT INTO RF_confusion_matrix VALUES (0, 1, 6217);
INSERT INTO RF_confusion_matrix VALUES (1, 0, 2571);
INSERT INTO RF_confusion_matrix VALUES (1, 1, 5412);
Commit;
```

```
DMU17>-- Insert values into the confusion matrix table
DMU17>INSERT INTO svm_confusion_matrix VALUES (0, 0, 17780);

1 row created.

DMU17>INSERT INTO svm_confusion_matrix VALUES (0, 1, 6217);

1 row created.

DMU17>INSERT INTO svm_confusion_matrix VALUES (1, 0, 2571);

1 row created.

DMU17>INSERT INTO svm_confusion_matrix VALUES (1, 1, 5412);

1 row created.

DMU17>commit;

Commit complete.

DMU17>/

Commit complete.

DMU17>select * from svm_confusion_matrix;

ACTUAL_TARGET_VALUE PREDICTED_TARGET_VALUE TOTAL_VALUE
------------------- ---------------------- -----------
                  0                      0       17780
                  0                      1        6217
                  1                      0        2571
                  1                      1        5412

DMU17>_
```

```
SET SERVEROUTPUT ON;
DECLARE
tp NUMBER;
fp NUMBER;
fn NUMBER;
tn NUMBER;
precision NUMBER;
recall NUMBER;
f1_score NUMBER;
accuracy NUMBER;
BEGIN
--Retrieve values from svm_confusion_matrix table
SELECT total_value INTO tp
FROM SVM_confusion_matrix
WHERE actual_target_value = 1 AND predicted_target_value = 1;
SELECT total_value INTO fp
FROM SVM_confusion_matrix
WHERE actual_target_value = 0 AND predicted_target_value = 1;
SELECT total_value INTO fn
FROM SVM_confusion_matrix
WHERE actual_target_value = 1 AND predicted_target_value = 0;
```

```
SELECT total_value INTO tn
FROM SVM_confusion_matrix
WHERE actual_target_value = 0 AND predicted_target_value = 0;
precision := tp / (tp + fp);
recall := tp / (tp + fn);
IF (precision + recall) = 0 THEN
f1_score := 0; -- Handle division by zero
ELSE
f1_score := 2 * (precision * recall) / (precision + recall);
END IF;
accuracy := (tp + tn) / (tp + fp + fn + tn);
DBMS_OUTPUT.PUT_LINE('True Positives (TP): ' || tp);
DBMS_OUTPUT.PUT_LINE('False Positives (FP): ' || fp);
DBMS_OUTPUT.PUT_LINE('False Negatives (FN): ' || fn);
DBMS_OUTPUT.PUT_LINE('True Negatives (TN): ' || tn);
DBMS_OUTPUT.PUT_LINE('Precision: ' || precision);
DBMS_OUTPUT.PUT_LINE('Recall: ' || recall);
DBMS_OUTPUT.PUT_LINE('F1 Score: ' || f1_score);
DBMS_OUTPUT.PUT_LINE('Accuracy: ' || accuracy);
END;
```

/

```
DMU17>SET SERVEROUTPUT ON;
DMU17>DECLARE
  tp NUMBER;
  fp NUMBER;
  fn NUMBER;
  tn NUMBER;
  precision NUMBER;
  recall NUMBER;
  f1_score NUMBER;
  accuracy NUMBER;
 BEGIN
 --Retrieve values from svm_confusion_matrix table
  SELECT total_value INTO tp
  FROM SVM_confusion_matrix
  WHERE actual_target_value = 1 AND predicted_target_value = 1;
  SELECT total_value INTO fp
  FROM SVM_confusion_matrix
  WHERE actual_target_value = 0 AND predicted_target_value = 1;
  SELECT total_value INTO fn
  FROM SVM_confusion_matrix
  WHERE actual_target_value = 1 AND predicted_target_value = 0;
  SELECT total_value INTO tn
  FROM SVM_confusion_matrix
  WHERE actual_target_value = 0 AND predicted_target_value = 0;
 precision := tp / (tp + fp);
  recall := tp / (tp + fn);
  IF (precision + recall) = 0 THEN
  f1_score := 0; -- Handle division by zero
  ELSE
  f1_score := 2 * (precision * recall) / (precision + recall);
  END IF;
  accuracy := (tp + tn) / (tp + fp + fn + tn);
  DBMS_OUTPUT.PUT_LINE('True Positives (TP): ' || tp);
  DBMS_OUTPUT.PUT_LINE('False Positives (FP): ' || fp);
  DBMS_OUTPUT.PUT_LINE('False Negatives (FN): ' || fn);
  DBMS_OUTPUT.PUT_LINE('True Negatives (TN): ' || tn);
  DBMS_OUTPUT.PUT_LINE('Precision: ' || precision);
  DBMS_OUTPUT.PUT_LINE('Recall: ' || recall);
  DBMS_OUTPUT.PUT_LINE('F1 Score: ' || f1_score);
  DBMS_OUTPUT.PUT_LINE('Accuracy: ' || accuracy);
 END;
 /
True Positives (TP): 5412
False Positives (FP): 6217
False Negatives (FN): 2571
True Negatives (TN): 17780
Precision: .4653882535041706079628514919597557829564
Recall: .6779406238256294626080420894400601277715
F1 Score: .5519069957169080155007138486640832143576
Accuracy: .7252032520325203252032520325203252

PL/SQL procedure successfully completed.
```

5. Present and critically discuss your findings and make recommendations to the Managing Director of the concerned company (iKHANiN or GLOBAL CREDIT CARDS).
   - Whereas SVM model generates higher volumes for false positives. It’s better to use Random Forest model even it has less volumes for true positives based on the cost cutting it is always better to avoid the model which has higher volumes for false positives.
   - Regularly monitoring the performance of both the models will increase in risk free on card issuance. Try to retain them with fresh data to ensure they improve the prediction capabilities.

- Random Forest model offers a better balance between detecting defaulters and minimum false alarms, it will be better to use in real world applications because of its F1 score.
- The certainty of any model is totally dependent on quality of dataset, try not to have the multiple occurrences such as columns names, duplicate values. Regular audits and check of the quality of the data can increase the accuracy and predictions
- If any sensitive data of customer such as pan number, ccv/ccv2 are used in dataset to train and test should be protected. Implement strict security measures to avoid data breach considering the privacy concerns of customer and adhere to company policies and work ethics.
- Consider using both models for different solutions. For example, use Random Forest model for precision, use SVM model for flag defaulters.

**Part 3**

*Critically evaluate the SH data warehouse and the two datasets related to the two scenarios of Appendix 2 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).

## Critical Evaluation of SH Data Warehouse and Two Datasets in Relation to Data Quality and Standards

### Introduction

Data has become very significant in the modern world in terms of how organizations go about their decision-making processes. GLOBAL CREDIT CARDS and iKHANiN Hotels have applied predictive analytics to transform challenges into opportunities. The SH data warehouse is a benchmark example, an easy point of reference to start understanding what data warehousing is like, because it has a star-schema based structure. However, in analytical efficiency, real-world complex data sets, like those of GLOBAL CREDIT CARDS and iKHANiN, have many limitations. Based on the datasets reviewed and the critical review of the SH data warehouse itself, gaps would be determined; data quality would be better while best practices have been adhered to. Various parts of Data Quality-accuracy, completeness, consistency, and timeliness-are reviewed in this report along with actionable recommendations for making improvements in performance and supporting decision-making outcomes.

### The SH Data Warehouse: Structure and Relevance

The SH schema is the star schema-based warehouse for data warehousing demonstrations, including one central fact table, SALES, and five dimension tables: TIMES, PRODUCTS, CUSTOMERS, CHANNELS, and PROMOTIONS. Its design focuses on analytical efficiency for fast aggregation and query execution. However, from a critical point of view, a manual review presents a number of strengths and limitations it faces:

1. **Strengths**
    - **Star Schema Design**: It enhances the efficiency of OLAP operations, and the analytics for sales are to be performed across many dimensions.
    - **Pre-aggregated Data**: Materialized views allow for improved query performance, especially for complex aggregations.

2. **Limitations**

    - **Volume and Variety of Data**: The schema supports moderate analytical tasks, but real-world datasets involve larger volumes and more complex relationships, hence limiting scalability.
    - **Lack of Granularity**: Some dimensions are not adequately granular to support finer degrees of personalization-a key requirement for GLOBAL CREDIT CARDS scenarios.

3. **Opportunities for Improvement**

    - **Time Series:** Expanding the dimension TIMES with a more granular time scale, for example, with hours, advanced forecasting models could be enabled.
    - **Integration of Further Fact Tables**: Adding auxiliary fact tables, such as revenues or refunds.

**Evaluation of Data Quality and Standards**

Data quality is fundamental for getting reliable insights from data mining and warehousing. The four key dimensions to evaluate are: accuracy, completeness, consistency, and timeliness.

1. **Accuracy:**
    - The GLOBAL CREDIT CARDS and iKHANiN datasets are highly accurate, as indicated by the structured and validated attributes in their data dictionaries.
    - However, inaccuracies in customer behavioural data-for instance, self-reported income levels-can make predictive models go awry. Inclusion of external validation mechanisms, like credit bureaus, would probably work much better.
2. **Completeness:**
    - The SH schema's SALES table provides comprehensive sales data, but lacks metadata for customer engagement or feedback, limiting its utility for customer retention models.
    - In GLOBAL CREDIT CARDS, the dataset lacks reasons for defaults and payment history patterns that might give a better understanding of customer behaviour.
3. **Consistency:**
    - The datasets exhibit structural consistency, with standardized field names and formats. However, maintaining consistency in derived attributes like risk scores (GLOBAL CREDIT CARDS) or cancellation probabilities (iKHANiN) requires robust ETL pipelines.
    - For SH schema, referential integrity with CUSTOMERS and between COUNTRIES should be enacted for consistent analysis along the dimensions.

4. **Timeliness:**
   - Both scenarios require real-time or near-real-time data to support predictive tasks; batch processing in the SH warehouse might delay actionable insights.
   - For GLOBAL CREDIT CARDS, the integration of streaming data pipelines of transactional data could improve timeliness and make risk assessments dynamic.

**Data Warehousing and Mining Best Practices**

1. **Dimensional Modelling:**
   - In GLOBAL CREDIT CARDS, the dimensional models can be developed based on customer demographics and payment behaviours to improve the segmentation strategies.
   - In iKHANiN, dimensions related to booking patterns and cancellation reasons will help in refining marketing efforts.

2. **Materialized Views and Indexing:**
   - Materialized views, as implemented in the SH warehouse, can be developed for GLOBAL CREDIT CARDS targeting high-frequency queries, such as customer default trends.
   - Indexing critical fields (e.g., cust_id, defaultnm) would decrease query costs and improve performance for predictive tasks.

3. **Data Mining Integration:**
   - The SH schema supports aggregation and transformation for data mining models. GLOBAL CREDIT CARDS can implement classification models (e.g., Random Forests) to predict defaults, while iKHANiN could leverage clustering for customer segmentation.

4. **Metadata Management:**
   - Both GLOBAL CREDIT CARDS and iKHANiN need a good metadata layer that could document data lineage, transformation, and definitions. In the SH schema, absence of metadata results in a structurally unextendable schema.

**Case-Specific Evaluation**

1. **GLOBAL CREDIT CARDS Dataset**:

   - Challenges: About 20% of customers default on payments, hence the need for strong predictive models.
   - Improvements: The addition of other factors, like credit utilization ratios or the number of missed payments, could provide a better model.

- Recommendations: The use of ensemble methods will be very effective, including Gradient Boosted Trees for high-performance predictions. Moreover, periodic model retraining is crucial for capturing changing customer behaviors.

2. **iKHANiN Hotel Bookings Dataset**:

   - Challenges: High cancellations (~33%) need Predictive Interventions.
   - Improvements: Capturing more customer attributes, such as loyalty program membership, and integrating external data, like weather forecasts, may further enhance the predictions.
   - Recommendations: Use regression models to quantify the impact of incentives and derive optimal cancellation policies.

**Recommendations and Future Directions**
Improving the quality and applicability of datasets is essential for GLOBAL CREDIT CARDS and iKHANiN to address their specific challenges and optimize decision-making. Below are the key recommendations and strategies for future directions:

1. Data Quality Enhancements

These can help in identifying anomalies during the ETL process earlier; hence, they can avoid issues later. The policy implements data governance policy uniformly so that it brings order and a very neat way of visualizing development and analysis into their studied subjects.

2. Advanced Analytics Adoption

For this, GLOBAL CREDIT CARDS can use the highly developed analytical software like deep learning methods, which is very apt for the most complex situations. Pioneer analytics can be used by iKHANiN to provide them, with granularity, significant data of their customers. This is very valuable mostly in the areas of fraud monitoring and automation.

3. Sustainability Considerations

Their ESG objectives, environmental, social, and governance practices might still align with these green computing targets when applied in data warehousing systems. Resource and energy consumption optimization through cloud exploitation will therefore minimize environmental impact per set of performance.

4. Customer-Centric Models

For mutual satisfaction, it is vital to incorporate feedback loops into the data workflows of both organizations. In the case of GLOBAL CREDIT CARDS, changes can arise regarding credit offerings through customer reflections, while for that of iKHANiN, details about personalized incentive and loyalty programs. Through both moves, better individualism will be assured with an increased loyalty status among customers.

**Conclusion**
About the SH Data Warehouse and datasets for GLOBAL CREDIT CARDS and iKhanIN show a rapidly growing phenomena due to a strong data quality, deep analytics and customer-centric approach on managing data. Although the SH schema is very strong foundational approach, it is limited when it comes to scalability and granularity in real world applications. Several more meaningful recommendations in GLOBAL CREDIT CARDS involve the removal of data quality issues and the

application of ensemble models to increase predictive power and customer retention. Predictive analytics and external data could be used in iKHANiN for improved cancellation management and revenue generation. The recommendation will make both organizations operationally excellent as well as aligned with sustainability goals while clearly delivering customer satisfaction in an increasingly data-driven landscape.

**References**:


Batini, C., Cappiello, C., Francalanci, C., and Maurino, A. (2009). Methodologies for data quality assessment and improvement. *ACM Computing Surveys, 41(3),* pp.1–52. doi:https://doi.org/10.1145/1541880.1541883.

Brown, J.S., Kahn, M., and Toh, S. (2013). Data Quality Assessment for Comparative Effectiveness Research in Distributed Data Networks. *Medical Care, 51(Supplement 8 Suppl 3),* pp.S22–S29. doi:https://doi.org/10.1097/mlr.0b013e31829b1e2c.

Das, S., Grbic, M., Ilic, I., Jovandic, I., Jovanovic, A., Narasayya, V.R., Radulovic, M., Stikic, M., Xu, G., and Chaudhuri, S. (2019). Automatically Indexing Millions of Databases in Microsoft Azure SQL Database. *Proceedings of the 2019 International Conference on Management of Data.* doi:https://doi.org/10.1145/3299869.3314035.

Goethals, F.G. (2002). Data warehousing and business intelligence for e-commerce. *ACM SIGMOD Record, 31(2),* pp.71–72. doi:https://doi.org/10.1145/565117.565132.

Khushairi, N., Emran, N., and Menon, A. (2018). Database Tuning Using Oracle Materialized View for Manufacturing Industry. International Journal of Computer Information Systems and Industrial Management Applications, 10, pp.38–46. Available at: https://www.mirlabs.org/ijcisim/regular_papers_2018/IJCISIM_5.pdf.

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

Pipino, L.L., Lee, Y.W., and Wang, R.Y. (2002). Data quality assessment. *Communications of the ACM, 45(4),* pp.211–218. doi:https://doi.org/10.1145/505248.506010.

Wongchinsri, P., and Kuratach, W. (2016). A survey - data mining frameworks in credit card processing. *Proceedings of the 2016 International Conference on Electronics and Communication Engineering.* doi:https://doi.org/10.1109/ecticon.2016.7561287.