```
use RETAIL;



CREATE OR REPLACE TABLE DEMOGRAPHIC_RAW(
        AGE_DESC CHAR(20),
        MARITAL_STATUS_CODE CHAR(5),
        INCOME_DESC VARCHAR(40),
  HOMEOWNER_DESC VARCHAR(50),
  HH_COMP_DESC VARCHAR(60),
  HOUSEHOLD_SIZE_DESC VARCHAR(20),
  KID_CATEGORY_DESC VARCHAR(80),
  household_key INT PRIMARY KEY

);

CREATE OR REPLACE TABLE CAMPAIGN_DESC_RAW(
  DESCRIPTION CHAR(10),
CAMPAIGN INT,
START_DAY INT,
END_DAY INT,
PRIMARY KEY (DESCRIPTION),
  UNIQUE (CAMPAIGN)
);

CREATE OR REPLACE TABLE CAMPAIGN_RAW(
DESCRIPTION CHAR(10),
household_key INT,
CAMPAIGN INT,
```

```
  FOREIGN KEY (DESCRIPTION) REFERENCES CAMPAIGN_DESC_RAW(DESCRIPTION),
  FOREIGN KEY (CAMPAIGN) REFERENCES CAMPAIGN_DESC_RAW(CAMPAIGN),
  FOREIGN KEY (household_key) REFERENCES DEMOGRAPHIC_RAW(household_key)
);

CREATE OR REPLACE TABLE PRODUCT_RAW(
PRODUCT_ID INT PRIMARY KEY,
MANUFACTURER INT,
DEPARTMENT VARCHAR(50),
BRAND VARCHAR(60),
COMMODITY_DESC VARCHAR(65),
SUB_COMMODITY_DESC VARCHAR(65),
CURR_SIZE_OF_PRODUCT VARCHAR(20)
);


CREATE OR REPLACE TABLE COUPON_RAW (
COUPON_UPC INT,
PRODUCT_ID INT,
CAMPAIGN INT,
FOREIGN KEY (CAMPAIGN) REFERENCES CAMPAIGN_DESC_RAW(CAMPAIGN),
  FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT_RAW(PRODUCT_ID)
);


CREATE OR REPLACE TABLE COUPON_REDEMPT_RAW (
household_key INT,
DAY INT,
COUPON_UPC INT,
```

```
CAMPAIGN INT,
FOREIGN KEY (CAMPAIGN) REFERENCES CAMPAIGN_DESC_RAW(CAMPAIGN),
  FOREIGN KEY (household_key) REFERENCES DEMOGRAPHIC_RAW(household_key)
);



CREATE OR REPLACE TABLE TRANSACTION_RAW (
household_key INT,
BASKET_ID INT,
DAY INT,
PRODUCT_ID INT,
QUANTITY INT,
SALES_VALUE FLOAT,
STORE_ID INT,
RETAIL_DISC FLOAT,
TRANS_TIME INT,
WEEK_NO INT,
COUPON_DISC INT,
COUPON_MATCH_DISC INT,
FOREIGN KEY (household_key) REFERENCES DEMOGRAPHIC_RAW(household_key),
FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT_RAW(PRODUCT_ID)
);



CREATE OR REPLACE STORAGE integration s3_int
TYPE = EXTERNAL_STAGE
STORAGE_PROVIDER = S3
ENABLED = TRUE
```

```
STORAGE_AWS_ROLE_ARN ='arn:aws:iam::991852474079:role/retailraw'

STORAGE_ALLOWED_LOCATIONS =('s3://ssretailraw/');


desc integration s3_int;


CREATE OR REPLACE STAGE retailraw

URL ='s3://ssretailraw'

--
credentials=(aws_key_id='AKIAXQKR3H3PSG72XFMK'aws_secret_key='eKL6a6FjlQHic4s8Ne712Aelzg2ou4j6tNsVvFq5')

file_format= csv_new

storage_integration =s3_int;




--CREATE SNOWPIPE THAT RECOGNISES CSV THAT ARE INGESTED FROM EXTERNAL STAGE AND COPIES THE DATA INTO EXISTING TABLE


--The AUTO_INGEST=true parameter specifies to read

--- event notifications sent from an S3 bucket to an SQS queue when new data is ready to load.



CREATE OR REPLACE PIPE RETAIL_SNOWPIPE_DEMOGRAPHIC AUTO_INGEST = TRUE AS

COPY INTO "RETAIL"."PUBLIC"."DEMOGRAPHIC_RAW" --yourdatabase -- your schema ---your table

FROM @retailraw/DEMOGRAPHIC/ --s3 bucket subfolde4r name

FILE_FORMAT = csv_new; --YOUR CSV FILE FORMAT NAME


CREATE OR REPLACE PIPE RETAIL_SNOWPIPE_CAMPAIGN_DESC AUTO_INGEST = TRUE AS

COPY INTO "RETAIL"."PUBLIC"."CAMPAIGN_DESC_RAW"

FROM @retailraw/CAMPAIGN_DESC/
```

```
FILE_FORMAT = csv_new;

CREATE OR REPLACE PIPE RETAIL_SNOWPIPE_CAMPAIGN AUTO_INGEST = TRUE AS
COPY INTO "RETAIL"."PUBLIC"."CAMPAIGN_RAW"
FROM @retailraw/CAMPAIGN/
FILE_FORMAT = csv_new;

CREATE OR REPLACE PIPE RETAIL_SNOWPIPE_PRODUCT AUTO_INGEST = TRUE AS
COPY INTO "RETAIL"."PUBLIC"."PRODUCT_RAW"
FROM @retailraw/PRODUCT/
FILE_FORMAT = csv_new;


CREATE OR REPLACE PIPE RETAIL_SNOWPIPE_COUPON AUTO_INGEST = TRUE AS
COPY INTO "RETAIL"."PUBLIC"."COUPON_RAW"
FROM @retailraw/COUPON/
FILE_FORMAT = csv_new;

CREATE OR REPLACE PIPE RETAIL_SNOWPIPE_COUPON_REDEMPT  AUTO_INGEST = TRUE AS
COPY INTO "RETAIL"."PUBLIC"."COUPON_REDEMPT_RAW"
FROM @retailraw/COUPON_REDEMPT/
FILE_FORMAT = csv_new;

CREATE OR REPLACE PIPE RETAIL_SNOWPIPE_TRANSACTION  AUTO_INGEST = TRUE AS
COPY INTO "RETAIL"."PUBLIC"."TRANSACTION_RAW"
FROM @retailraw/TRANSACTION/
FILE_FORMAT = csv_new;

SHOW PIPES;
```

```
drop table transaction_raw;


SELECT COUNT(*) FROM demographic_RAW;
SELECT COUNT(*) FROM CAMPAIGN_DESC_RAW;
SELECT COUNT(*) FROM CAMPAIGN_RAW;
SELECT COUNT(*) FROM PRODUCT_RAW;
SELECT COUNT(*) FROM COUPON_RAW;
SELECT COUNT(*) FROM COUPON_REDEMPT_RAW;
SELECT COUNT(*) FROM TRANSACTION_RAW;

------------------------------------------------------------PIPEREFRESH-----------------------------------------------------
--


ALTER PIPE RETAIL_SNOWPIPE_DEMOGRAPHIC refresh;
ALTER PIPE  RETAIL_SNOWPIPE_CAMPAIGN_DESC refresh;
ALTER PIPE  RETAIL_SNOWPIPE_CAMPAIGN refresh;
ALTER PIPE  RETAIL_SNOWPIPE_PRODUCT refresh;
ALTER PIPE  RETAIL_SNOWPIPE_COUPON refresh;
ALTER PIPE  RETAIL_SNOWPIPE_COUPON_REDEMPT refresh;
ALTER PIPE  RETAIL_SNOWPIPE_TRANSACTION refresh;

CREATE OR REPLACE PROCEDURE ALTER_TXN_PIPE()
RETURNS STRING
LANGUAGE SQL
AS
$$
ALTER PIPE  RETAIL_SNOWPIPE_TRANSACTION refresh;
$$;
```

```
CREATE OR REPLACE TASK ALTER_TXN_PIPE
WAREHOUSE = COMPUTE_WH
SCHEDULE = 'USING CRON 00 20 * * FRI Asia/Kolkata'
AS CALL ALTER_TXN_PIPE();

ALTER TASK   ALTER_TXN_PIPE RESUME;
ALTER TASK  ALTER_TXN_PIPE SUSPEND;




SELECT * FROM demographic_RAW;
SELECT * FROM CAMPAIGN_DESC_RAW;
SELECT * FROM CAMPAIGN_RAW;
SELECT * FROM PRODUCT_RAW;
SELECT * FROM COUPON_RAW;
SELECT * FROM COUPON_REDEMPT_RAW;
SELECT * FROM TRANSACTION_RAW;

LIST @retailraw;

SHOW STAGES;
---------------------------------------------- NEW TABLES ADDED FROM PYTHON

select * from CAMPAIGN_DESC_NEW;

select * from COUPON_REDEMPT_NEW;
```

```
select * from TRANSACTION_NEW;


-----deparment wise product count

SELECT DISTINCT(DEPARTMENT),COUNT(*) AS TOTAL_PRODUCT

FROM PRODUCT_RAW

GROUP BY 1

ORDER BY 2 DESC;



/*1.	Customer Demographics KPIs:

A. Count of unique households: Measure the total number of unique households in the Demographic table.

B. Household composition distribution: Analyze the distribution of household compositions (HH_COMP_DESC) to understand the composition of households.

C.	Age distribution: Calculate the percentage or count of customers in different age groups (AGE_DESC).

D.	Marital status distribution: Analyze the proportion of customers in different marital status categories (MARITAL_STATUS_CODE).

E.	Income distribution: Determine the distribution of customers across income levels (INCOME_DESC).

F. Homeownership distribution: Calculate the percentage or count of customers who own or rent their homes (HOMEOWNER_DESC).*/


SELECT COUNT(DISTINCT HOUSEHOLD_KEY) AS TOTAL_HOUSEHOLDS FROM DEMOGRAPHIC_RAW; --2,500


SELECT HH_COMP_DESC,COUNT(DISTINCT HOUSEHOLD_KEY) AS TOTAL_HOUSEHOLDS

FROM DEMOGRAPHIC_RAW

GROUP BY 1

ORDER BY 2 DESC;
```

```
SELECT AGE_DESC,TOTAL_HOUSEHOLDS,ROUND(TOTAL_HOUSEHOLDS/2500 * 100,2) AS
PERC_AGEWISE_HOUSEHOLDS_DISTR

FROM

(SELECT AGE_DESC,

COUNT(DISTINCT HOUSEHOLD_KEY) AS TOTAL_HOUSEHOLDS

FROM demographic_RAW

GROUP BY 1

ORDER BY 2 DESC)

GROUP BY 1,2;


SELECT MARITAL_STATUS_CODE ,

COUNT(DISTINCT HOUSEHOLD_KEY) AS TOTAL_HOUSEHOLDS,

ROUND(COUNT(DISTINCT HOUSEHOLD_KEY) / 2500 * 100 , 2) AS PERC_MARITAL_HOUSEHOLDS_DISTR

FROM demographic_RAW

GROUP BY 1

ORDER BY 2 DESC;




SELECT INCOME_DESC ,

COUNT(DISTINCT HOUSEHOLD_KEY) AS TOTAL_HOUSEHOLDS,

ROUND(COUNT(DISTINCT HOUSEHOLD_KEY) / 2500 * 100 , 2) AS PERC_INCOME_HOUSEHOLDS_DISTR

FROM demographic_RAW

GROUP BY 1

ORDER BY 2 DESC;



SELECT HOMEOWNER_DESC ,
```

```
COUNT(DISTINCT HOUSEHOLD_KEY) AS TOTAL_HOUSEHOLDS,

ROUND(COUNT(DISTINCT HOUSEHOLD_KEY) / 2500 * 100 , 2) AS PERC_HOMEOWNER_DESC_DISTR

FROM demographic_RAW

GROUP BY 1

ORDER BY 2 DESC;



SELECT
T.HOUSEHOLD_KEY,D.AGE_DESC,D.MARITAL_STATUS_CODE,D.INCOME_DESC,AVG(T.SALES_VALUE)AS
AVG_AMOUNT,

AVG(T.RETAIL_DISC)AS AVG_RETAIL_DIS,AVG(T.COUPON_DISC)AS
AVG_COUPON_DISC,AVG(T.COUPON_MATCH_DISC)AS AVG_COUP_MATCH_DISC

FROM TRANSACTION_NEW T

LEFT OUTER JOIN demographic_RAW D ON T.HOUSEHOLD_KEY =D.HOUSEHOLD_KEY

GROUP BY 1,2,3,4

ORDER BY 1;



CREATE OR REPLACE PROCEDURE Household_kpi()

RETURNS STRING

LANGUAGE SQL

AS

$$

 CREATE OR REPLACE TABLE Household_kpi AS (SELECT
T.HOUSEHOLD_KEY,D.AGE_DESC,D.MARITAL_STATUS_CODE,D.INCOME_DESC,AVG(T.SALES_VALUE)AS
AVG_AMOUNT,

AVG(T.RETAIL_DISC)AS AVG_RETAIL_DIS,AVG(T.COUPON_DISC)AS
AVG_COUPON_DISC,AVG(T.COUPON_MATCH_DISC)AS AVG_COUP_MATCH_DISC

FROM TRANSACTION_NEW T

LEFT OUTER JOIN demographic_RAW D ON T.HOUSEHOLD_KEY = D.HOUSEHOLD_KEY
```

```
GROUP BY 1,2,3,4

ORDER BY 1);

$$;


SHOW PROCEDURES;


CALL Household_kpi();


CREATE OR REPLACE TASK  Household_kpi_TASK

WAREHOUSE = COMPUTE_WH

SCHEDULE = 'USING CRON 20 20 * * FRI Asia/Kolkata'

AS CALL Household_kpi();


SHOW TASKS;


ALTER TASK   Household_kpi_TASK RESUME;

ALTER TASK  Household_kpi_TASK SUSPEND;



--2. Campaign KPIs:

-- Number of campaigns: Count the total number of campaigns in the Campaign table.

--

-- Campaign duration: Calculate the duration of each campaign by subtracting the start day from the end day (in the Campaign_desc table).

--

-- Campaign effectiveness: Analyze the number of households associated with each  campaign (in the Campaign table) to measure campaign reach.

select * from CAMPAIGN_DESC_NEW;

select * from CAMPAIGN_RAW;
```

```
SELECTRETAIL.PUBLIC.CAMPAIGN_DESC_NEW * FROM CAMPAIGN_DESC_NEW;


--1

select count(distinct campaign ) from CAMPAIGN_RAW;


--2

select campaign, start_date, end_date, campaign_duration

from CAMPAIGN_DESC_NEW

order by campaign_duration desc;



--3

select campaign, count(distinct household_key) as TOTAL_HOUSEHOLDS

from CAMPAIGN_RAW

group by 1

order by 2 desc;

----------------------------------------------     CAMPAIGN KPI TABLE    ----------------------------------

CREATE OR REPLACE PROCEDURE CAMPAIGN_KPI()

RETURNS STRING

LANGUAGE SQL

AS

$$

 CREATE OR REPLACE TABLE CAMPAIGN_KPI AS SELECT CR.CAMPAIGN, COUNT(DISTINCT
TXN.HOUSEHOLD_KEY), SUM (TXN.SALES_VALUE) AS TOTAL_SALES,

 CASE WHEN CR.CAMPAIGN IS NULL THEN 'NO_CAMPAIGN_TXN' ELSE 'CAMPAIGN_TXN' END AS
CAMPAIGN_STATUS

FROM TRANSACTION_NEW AS TXN LEFT OUTER JOIN CAMPAIGN_RAW AS CR USING(HOUSEHOLD_KEY)

GROUP BY 1
```

```
ORDER BY 2 DESC, 3 DESC;
$$;

CREATE OR REPLACE TASK CAMPAIGN_KPI_TASK
WAREHOUSE = COMPUTE_WH
SCHEDULE = 'USING CRON 26 20 * * FRI Asia/Kolkata'
AS CALL CAMPAIGN_KPI();

SHOW TASKS;

ALTER TASK   CAMPAIGN_KPI_TASK RESUME;
ALTER TASK  CAMPAIGN_KPI_TASK SUSPEND;

-------------------------------------------------------------------------------------------------------------------------------
--
-- IMPACT OF CAMPAIGN ON SALES

-- SELECT CR.CAMPAIGN, COUNT(DISTINCT TXN.HOUSEHOLD_KEY)AS COUNT_HOUSEHOLDS,
SUM(TXN.SALES_VALUE), AVG(TXN.SALES_VALUE)
-- FROM TRANSACTION_NEW AS TXN LEFT OUTER JOIN CAMPAIGN_RAW AS CR
USING(HOUSEHOLD_KEY)
-- GROUP BY 1
-- ORDER BY 4 DESC, 2 DESC, 3 DESC;

-- SELECT CAMPAIGN_STATUS, AVG(TOTAL_SALES) AS AVG_SALES FROM (
```

```
-- SELECT CR.CAMPAIGN, COUNT(DISTINCT TXN.HOUSEHOLD_KEY), SUM(TXN.QUANTITY*
(TXN.SALES_VALUE + TXN.RETAIL_DISC )) AS TOTAL_SALES, CASE WHEN CR.CAMPAIGN IS NULL THEN
'NO_CAMPAIGN_TXN' ELSE 'CAMPAIGN_TXN' END AS CAMPAIGN_STATUS

-- FROM TRANSACTION_NEW AS TXN LEFT OUTER JOIN CAMPAIGN_RAW AS CR
USING(HOUSEHOLD_KEY)

-- GROUP BY 1

-- ORDER BY 2 DESC,3 DESC)

-- GROUP BY 1;



-- SELECT DISTINCT CR.CAMPAIGN,  TXN.HOUSEHOLD_KEY, TXN.SALES_VALUE, TXN.SALES_VALUE

-- FROM TRANSACTION_NEW AS TXN LEFT OUTER JOIN CAMPAIGN_RAW AS CR
USING(HOUSEHOLD_KEY)

-- ORDER BY 4 DESC, 2 DESC, 3 DESC;



---3. Coupon KPIs:

-- Coupon redemption rate: Calculate the percentage of coupons redeemed (from the coupon_redempt
table) compared to the total number of coupons distributed (from the Coupon table).

-- Coupon usage by campaign: Measure the number of coupon redemptions (from the coupon_redempt
table) for each campaign (in the Coupon table).



select * from "RETAIL"."PUBLIC"."COUPON_REDEMPT_NEW";


select * from "RETAIL"."PUBLIC"."COUPON_RAW";



--1 --- IN ORDER TO RETREIVE ONLY THE PERCENTAGE
```

```
select campaign,total_given, total_redeempt,  round((total_redeempt/total_given) *100,2) as perent
from

(select campaign, count(distinct COUPON_UPC) as total_redeempt from COUPON_REDEMPT_NEW
group by 1) a join

(select campaign, count(distinct COUPON_UPC) as total_given from COUPON_RAW group by 1) b
using(campaign);



--select campaign, round((total_redeempt/cr.total_given) *100,2) as percent



select CAMPAIGN, COUNT(DISTINCT PRODUCT_ID) AS TOTAL_PRODUCTS,count(distinct COUPON_UPC)
as total_count

from COUPON_RAW

group by 1

order by 2 desc;


--- 1ST APPROACH

select CAMPAIGN, NCR.TOTAL_PRODUCTS, NCR.TOTAL_COUPON_GIVEN, count(distinct COUPON_UPC)
as TOTAL_COUPON_REDEEM , ROUND((TOTAL_COUPON_REDEEM/TOTAL_COUPON_GIVEN)*100, 2) AS
PERCENT

from COUPON_REDEMPT_NEW

INNER JOIN (select CAMPAIGN, COUNT(DISTINCT PRODUCT_ID) AS TOTAL_PRODUCTS,count(distinct
COUPON_UPC) as TOTAL_COUPON_GIVEN

from COUPON_RAW

group by 1) AS NCR USING (CAMPAIGN)

group by 1,2,3

order by 5 desc;
```

```
--2D APPROACH FOR 1ST QUES

select sum(total_count) from (

SELECT CAMPAIGN, COUNT(DISTINCT PRODUCT_ID) AS TOTAL_PRODUCTS, COUNT(DISTINCT
CR.COUPON_UPC) as TOTAL_COUNT, CRN.TOTA_REDEM_COUPON

FROM COUPON_RAW AS CR LEFT JOIN

(SELECT CAMPAIGN, COUNT(DISTINCT COUPON_UPC) AS TOTA_REDEM_COUPON FROM
COUPON_REDEMPT_NEW GROUP BY 1) AS CRN USING(CAMPAIGN)

group by 1,4

order by 2 desc);



--2D ANS SAME AS 1ST


select CAMPAIGN, COUNT(DISTINCT PRODUCT_ID) AS TOTAL_PRODUCTS,count(distinct COUPON_UPC)
as TOTAL_COUNT, CRN.TOTA_REDEM_COUPON

from COUPON_RAW LEFT JOIN

(SELECT CAMPAIGN, COUNT(DISTINCT COUPON_UPC) AS TOTA_REDEM_COUPON FROM
COUPON_REDEMPT_NEW GROUP BY 1) AS CRN USING(CAMPAIGN)

group by 1,4

order by 2 desc;


select count(distinct commodity_desc) from coupon_raw join product_raw using(product_id);

---------- COUPONS REEDEMED PER PRODUCT


SELECT DISTINCT CR.PRODUCT_ID AS PRODUCT_IDS , PR.COMMODITY_DESC AS COMMODITY,
CRN.CAMPAIGN, CRN.COUPON_UPC  AS REDEEMED_COUPONS

FROM COUPON_REDEMPT_NEW AS CRN JOIN COUPON_RAW AS CR USING(COUPON_UPC)

 JOIN PRODUCT_RAW AS PR ON PR.PRODUCT_ID = CR.PRODUCT_ID

GROUP BY 1

ORDER BY 2 DESC;
```

---------- COUPON_KPI_TABLE ----------

```
CREATE OR REPLACE PROCEDURE COUPON_KPI()
RETURNS STRING
LANGUAGE SQL
AS
$$
 CREATE OR REPLACE TABLE COUPON_KPI AS select CAMPAIGN, NCR.TOTAL_PRODUCTS, NCR.TOTAL_COUPON_GIVEN, count(distinct COUPON_UPC) as TOTAL_COUPON_REDEEM , ROUND((TOTAL_COUPON_REDEEM/TOTAL_COUPON_GIVEN)*100, 2) AS PERCENT
from COUPON_REDEMPT_NEW
INNER JOIN (select CAMPAIGN, COUNT(DISTINCT PRODUCT_ID) AS TOTAL_PRODUCTS,count(distinct COUPON_UPC) as TOTAL_COUPON_GIVEN
from COUPON_RAW
group by 1) AS NCR USING (CAMPAIGN)
group by 1,2,3
order by 5 desc;


$$;
```

```
CREATE OR REPLACE TASK COUPON_KPI_TASK
WAREHOUSE = COMPUTE_WH
SCHEDULE = 'USING CRON 32 20 * * FRI Asia/Kolkata'
AS CALL COUPON_KPI();
```

```
SHOW TASKS;

ALTER TASK   COUPON_KPI_TASK RESUME;
ALTER TASK  COUPON_KPI_TASK SUSPEND;




select store_id, count(distinct household_key), count(distinct crn.coupon_upc) AS COUP_COUNT
from transaction_new LEFT join coupon_redempt_new as crn using(household_key)
group by 1
HAVING COUP_COUNT = 0
order by 2 DESC;

select  txn.household_key, count(distinct crn.coupon_upc)
from transaction_new txn join coupon_redempt_new as crn using(household_key)
group by 1;

select household_key, count(distinct coupon_upc)
from coupon_redempt_new group by 1;


select crn.coupon_upc, count(distinct household_key) as total
from transaction_new join coupon_redempt_new as crn using(household_key)
where coupon_status = 'Coupon Used'
group by 1;
```

```
select coupon_status, count(distinct household_key)

from transaction_new group by 1;



-- 4 Product KPIs:

-- Sales value: Calculate the total sales value for each product (in the Transaction_data table) to identify top-selling products.

-- Manufacturer distribution: Analyze the distribution of products across different manufacturers (in the Product table).

-- Department-wise sales: Measure the sales value by department (in the Product table) to understand which departments contribute most to revenue.

-- Brand-wise sales: Calculate the sales value for each brand (in the Product table) to identify top-selling brands.


SELECT * FROM PRODUCT_RAW

WHERE PRODUCT_ID = 26093;

SELECT * FROM TRANSACTION_NEW;



SELECT  COUNT(DISTINCT SUB_COMMODITY_DESC) FROM PRODUCT_RAW;


SELECT COUNT(DISTINCT PRODUCT_ID) FROM TRANSACTION_NEW;


-- 1

SELECT PRODUCT_ID, PR.COMMODITY_DESC , SUM(QUANTITY* (SALES_VALUE + RETAIL_DISC )) AS TOTAL_SALES

FROM TRANSACTION_NEW JOIN

PRODUCT_RAW AS PR USING(PRODUCT_ID)

GROUP BY 1,2

ORDER BY 3 DESC;
```

```
-- 2
SELECT MANUFACTURER, COUNT(PRODUCT_ID) AS TOTAL_PRODUCTS
FROM PRODUCT_RAW
SAMPLE (100)
GROUP BY 1
ORDER BY 2 DESC
;
SELECT * FROM PRODUCT_RAW
SAMPLE (50)
;

-- 3
SELECT * FROM
(
SELECT PR.DEPARTMENT , SUM(QUANTITY* (SALES_VALUE + RETAIL_DISC )) AS TOTAL_SALES
FROM TRANSACTION_NEW  JOIN
PRODUCT_RAW AS PR USING(PRODUCT_ID)
--WHERE PR.DEPARTMENT NOT IN ('GROCERY','PRODUCE')
GROUP BY 1
ORDER BY 2 DESC)
SAMPLE (50);

-- 4
SELECT PR.BRAND , SUM(QUANTITY* (SALES_VALUE + RETAIL_DISC )) AS TOTAL_SALES
FROM TRANSACTION_NEW JOIN
PRODUCT_RAW AS PR USING(PRODUCT_ID)
GROUP BY 1
```

```
ORDER BY 2 DESC;
```

---------- PRODUCT KPI TABLE ----------

--- we could have Also ued used inner join in this scenario as that will give the same result as we got from left outer join

```
CREATE OR REPLACE PROCEDURE Product_kpi()
RETURNS STRING
LANGUAGE SQL
AS
$$
 CREATE OR REPLACE TABLE Product_kpi AS SELECT PRODUCT_ID, PR.MANUFACTURER, PR.COMMODITY_DESC, PR.DEPARTMENT, PR.BRAND , SUM(SALES_VALUE) AS TOTAL_SALES
FROM TRANSACTION_NEW LEFT OUTER JOIN
PRODUCT_RAW AS PR USING(PRODUCT_ID)
GROUP BY 1,2,3,4,5
ORDER BY 2 DESC;
$$;

CREATE OR REPLACE TASK PRODUCT_KPI_TASK
WAREHOUSE = COMPUTE_WH
SCHEDULE = 'USING CRON 38 20 * * FRI Asia/Kolkata'
AS CALL Product_kpi();

SHOW TASKS;
```

```
ALTER TASK   PRODUCT_KPI_TASK RESUME;

ALTER TASK  PRODUCT_KPI_TASK SUSPEND;




-- 5. Transaction KPIs:


-- Total sales value: Calculate the sum of sales values (in the Transaction_data table)to measure overall revenue.


-- Average transaction value: Calculate the average sales value per transaction to understand customer spending patterns.


-- Quantity sold: Measure the total quantity sold (in the Transaction_data table) to understand product demand.


-- Discounts: Analyze the amount and impact of discounts (RETAIL_DISC, COUPON_DISC, COUPON_MATCH_DISC) on sales value.



SELECT * FROM TRANSACTION_NEW

WHERE COUPON_MATCH_DISC!=0 ;


-- 1 COLUMN CEATED AS TOTAL_REVENUE


SELECT *, QUANTITY * (SALES_VALUE + RETAIL_DISC) AS TOTAL_REVENUE FROM TRANSACTION_NEW;


-- 2  AVG SALES PE BASKET


SELECT ROUND(AVG(SALES_VALUE),2) AS AVG_SALES_PER_BASKET FROM
```

```
(
    SELECT BASKET_ID, ROUND(SUM (SALES_VALUE),2) AS SALES_VALUE
    FROM TRANSACTION_NEW
    GROUP BY 1
    ORDER BY 2 DESC
);

--SELECT COUNT(DISTINCT BASKET_ID) FROM TRANSACTION_NEW;  -- 50479

--SUM SALES_VALUE PER BASKET AFTER DISC
SELECT BASKET_ID, ROUND(SUM(QUANTITY * (SALES_VALUE + RETAIL_DISC)),2) AS AVG_REVENUE
FROM TRANSACTION_NEW
GROUP BY 1
ORDER BY 2 DESC;

-- 3

SELECT PRODUCT_ID, SUM(QUANTITY) AS SUM_REVENUE
FROM TRANSACTION_NEW
GROUP BY 1
ORDER BY 2 DESC;

-- 4
SELECT  case when COUPON_DISC <> 0 then 'Coupon used' else 'Coupon not used' end as COUP_STATUS
, avg(SALES), AVG(COUPON_DISC), AVG(COUPON_MATCH_DISC), COUNT(SALES) FROM (
```

```
SELECT  BASKET_ID,  SUM(SALES_VALUE) AS SALES, SUM(COUPON_DISC)AS COUPON_DISC ,
SUM(COUPON_MATCH_DISC) AS COUPON_MATCH_DISC

FROM TRANSACTION_NEW

GROUP BY 1

order by 2 desc)

GROUP BY 1

;



SELECT  BASKET_ID, COUPON_STATUS, SUM(SALES_VALUE) AS SALES, SUM(COUPON_DISC)AS
COUPON_DISC , SUM(COUPON_MATCH_DISC) AS COUPON_MATCH_DISC

FROM TRANSACTION_NEW

GROUP BY 1,2

order by 2 desc;


--- PRODUCT FREQUENTLY PURCHASED TOGETHER

-- SELECT T1.PRODUCT_ID, T2.PRODUCT_ID, COUNT(*) AS TOTAL

-- FROM TRANSACTION_NEW T1

-- INNER JOIN TRANSACTION_NEW T2

-- ON T1.HOUSEHOLD_KEY = T2.HOUSEHOLD_KEY AND T1.BASKET_ID = T2.BASKET_ID AND
T1.PRODUCT_ID < T2.PRODUCT_ID

-- GROUP BY 1,2

-- ORDER BY 3 DESC;


CREATE OR REPLACE TABLE PRODUCTS_BOUGHT_TOGETHER AS (


   SELECT T1.PRODUCT_ID AS PRODUCT_ONE, T1.SUB_COMMODITY_DESC AS DESCRIPTION_ONE,
T2.PRODUCT_ID AS PRODUCT_TWO, T2.SUB_COMMODITY_DESC AS DESCRIPTION_TWO, COUNT(*) AS
TOTAL

   FROM (SELECT HOUSEHOLD_KEY, PRODUCT_ID, BASKET_ID, PR.SUB_COMMODITY_DESC
```

```
   FROM TRANSACTION_NEW LEFT OUTER JOIN
   PRODUCT_RAW AS PR USING(PRODUCT_ID)
   ORDER BY 2 DESC) T1

INNER JOIN
   (SELECT HOUSEHOLD_KEY, PRODUCT_ID, BASKET_ID, PR.SUB_COMMODITY_DESC
   FROM TRANSACTION_NEW LEFT OUTER JOIN
   PRODUCT_RAW AS PR USING(PRODUCT_ID)
   ORDER BY 2 DESC) T2

ON T1.HOUSEHOLD_KEY = T2.HOUSEHOLD_KEY AND T1.BASKET_ID = T2.BASKET_ID AND
T1.PRODUCT_ID < T2.PRODUCT_ID
GROUP BY 1,2,3,4
ORDER BY 5 DESC);

-- SELECT HOUSEHOLD_KEY, PRODUCT_ID, PR.MANUFACTURER, PR.COMMODITY_DESC,
PR.DEPARTMENT, PR.BRAND , SUM(SALES_VALUE ) AS TOTAL_SALES
-- FROM TRANSACTION_NEW LEFT OUTER JOIN
-- PRODUCT_RAW AS PR USING(PRODUCT_ID)
-- GROUP BY 1,2,3,4,5,6
-- ORDER BY 2 DESC;


select count( coupon_upc) from coupon_raw;

SELECT HOUSEHOLD_KEY, PR.COMMODITY_DESC, COUNT(TXN.PRODUCT_ID)
FROM TRANSACTION_NEW AS TXN JOIN PRODUCT_RAW AS PR USING(PRODUCT_ID)
GROUP BY 1,2
ORDER BY 3 DESC;
```