JSON data:-

CREATE OR REPLACE TABLE staging.employee\_data (

data VARIANT

);

INSERT INTO staging.employee\_data (data)

SELECT PARSE\_JSON('{"Name": "Sai", "Gender": "Male", "Skills": ["Snowflake", "DBT"], "Role": "Data Engineer"}')

UNION ALL

SELECT PARSE\_JSON('{"Name": "Kishore", "Gender": "Male", "Skills": ["Java", "Spring boot"], "Role": "Frontend Developer"}')

UNION ALL

SELECT PARSE\_JSON('{"Name": "Manoj", "Gender": "Male", "Skills": ["Snowflake", "Power BI"], "Role": "Analytics Engineer"}')

UNION ALL

SELECT PARSE\_JSON('{"Name": "Uday", "Gender": "Male", "Skills": ["Alteryx", "SQL"], "Role": "Data Analyst"}');

select \* from staging.employee\_data;

CREATE or replace table public.skills(

Name Varchar,

Gender Varchar,

skills array,

"Role" varchar

);

INSERT INTO public.skills (Name, Gender, Skills, "Role")

SELECT

data:Name::Varchar AS Name,

data:Gender::Varchar AS Gender,

data:Skills::ARRAY AS Skills,

data:Role::Varchar AS Role

FROM staging.employee\_data;

SELECT \*

FROM public.skills

WHERE ARRAY\_CONTAINS('Alteryx'::VARIANT, Skills) or

"Role" = 'Data Engineer';

Procedures:-

CREATE OR REPLACE PROCEDURE MY\_DATABASE.PROCEDURES.IMDB\_multi(YEAR Varchar, imdb\_score number)

RETURNS TABLE (MOVIE Varchar, IMDB\_SCORE number,year varchar)

LANGUAGE SQL

EXECUTE AS CALLER

AS $$

BEGIN

-- Create a result set

LET result RESULTSET := (

SELECT movie::varchar, imdb\_score::number,year::varchar

FROM public.marvell\_dc\_data

WHERE public.marvell\_dc\_data.year = :YEAR and public.marvell\_dc\_data.imdb\_score = :imdb\_score

);

RETURN TABLE (result);

END;

$$;

call procedures.imdb\_multi('2021',8.0);

Create or Replace procedure mobile\_data(BRAND varchar,stars number)

Returns TABLE(PRODUCT\_NAME varchar,ACTUAL\_PRICE varchar, DISCOUNT\_PRICE varchar,DISCOUNT\_PERCENT number,DESCRIPTION varchar,stars float)

LANGUAGE SQL

EXECUTE AS CALLER

as

$$

begin

LET result\_table resultset := (

select PRODUCT\_NAME:: Varchar , ACTUAL\_PRICE::Varchar , DISCOUNT\_PRICE :: Varchar, DISCOUNT\_PERCENT :: Number , DESCRIPTION :: Varchar, stars :: float

from my\_database.public.mobile\_discount

where brand\_name =:brand and stars >=:stars

);

return table(result\_table);

end;

$$;

call mobile\_data('I Kall', 3);

CREATE OR REPLACE PROCEDURE movie\_detail(p\_genre VARCHAR)

RETURNS TABLE (movie VARCHAR, year VARCHAR, genre VARCHAR, IMDB\_SCORE NUMBER(38,1), movie\_rating VARCHAR)

LANGUAGE SQL

EXECUTE AS CALLER

AS

$$

BEGIN

LET result resultset:= (

SELECT movie, year, genre, IMDB\_SCORE,

CASE

WHEN IMDB\_SCORE > 4.0 THEN 'Excellent'

WHEN IMDB\_SCORE > 3.0 AND IMDB\_SCORE <= 4.0 THEN 'Good'

WHEN IMDB\_SCORE > 2.0 AND IMDB\_SCORE <= 3.0 THEN 'Average'

ELSE 'Poor'

END AS movie\_rating

FROM my\_database.public.marvell\_dc\_data

WHERE contains(genre,:p\_genre)

);

return table(result);

END;

$$;

Call movie\_detail('Fantasy');

CREATE OR REPLACE PROCEDURE mobiles\_sales\_amount(p\_brand VARCHAR)

RETURNS STRING

LANGUAGE SQL

EXECUTE AS CALLER

AS

$$

DECLARE

cur CURSOR FOR

SELECT brand\_name,sum(DISCOUNT\_PRICE) as DISCOUNT\_PRICE

FROM my\_database.public.mobile\_discount

WHERE upper(brand\_name) = upper(?)

GROUP BY brand\_name;

brand VARCHAR;

amount number;

models number;

BEGIN

OPEN cur using(:p\_brand);

FETCH cur INTO brand, amount;

CLOSE cur;

RETURN 'The total amount of ' || brand || ' is ' || amount ;

END;

$$;

call mobiles\_sales\_amount('lava');

CREATE OR REPLACE PROCEDURE mobile\_rank(BRAND\_NAME VARCHAR)

RETURNS TABLE(PRODUCT\_NAME VARCHAR, STARS FLOAT, DISCOUNT\_PRICE NUMBER, DESCRIPTION VARCHAR, DISCOUNT\_PERCENT NUMBER, RANKING INT)

LANGUAGE SQL

EXECUTE AS CALLER

AS

$$

Begin

LET res resultset :=(

SELECT \*

FROM (

SELECT PRODUCT\_NAME :: VARCHAR, STARS :: FLOAT, DISCOUNT\_PRICE :: NUMBER, DESCRIPTION :: VARCHAR, DISCOUNT\_PERCENT :: NUMBER,

DENSE\_RANK() OVER (PARTITION BY BRAND\_NAME ORDER BY STARS DESC, DISCOUNT\_PERCENT ASC) AS ranking

FROM my\_database.public.mobile\_discount

WHERE UPPER(BRAND\_NAME) = UPPER(:BRAND\_NAME)

)

WHERE ranking = 1

);

Return table(res);

END;

$$;

Call MOBILE\_RANK('Apple');

Exceptions:-

CREATE OR REPLACE PROCEDURE mobile\_exception(BRAND\_NAME VARCHAR)

RETURNS int

LANGUAGE SQL

EXECUTE AS CALLER

AS

DECLARE

brand\_count INT := 0;

no\_data\_found EXCEPTION (-20101, 'Brand does not exist');

BEGIN

SELECT COUNT(BRAND\_NAME) into :brand\_count

FROM my\_database.public.mobile\_discount

WHERE upper(BRAND\_NAME) = upper(:BRAND\_NAME);

--GROUP BY BRAND\_NAME;

IF (brand\_count = 0) THEN

RAISE no\_data\_found;

END IF;

return brand\_count;

EXCEPTION

WHEN statement\_error THEN

insert into exception\_table("procedure name" ,"Error type" ,"SQL CODE" ,"SQL ERRM" ,"SQL STATE" )

values('mobile\_exception','statement\_error',:SQLCODE,:SQLERRM,:SQLSTATE);

return 0;

WHEN expression\_error THEN

insert into exception\_table("procedure name" ,"Error type" ,"SQL CODE" ,"SQL ERRM" ,"SQL STATE" )

values('mobile\_exception','expression\_error',:SQLCODE,:SQLERRM,:SQLSTATE);

return 0;

WHEN no\_data\_found THEN

insert into exception\_table("procedure name" ,"Error type" ,"SQL CODE" ,"SQL ERRM" ,"SQL STATE" )

values('mobile\_exception','User-Defined Exception',:SQLCODE,:SQLERRM,:SQLSTATE);

return 0;

END;

CREATE TABLE IF NOT EXISTS exception\_table (

"procedure name" VARCHAR,

"Error type" VARCHAR,

"SQL CODE" VARCHAR,

"SQL ERRM" VARCHAR,

"SQL STATE" VARCHAR

);

select \* from exception\_table;

Call mobile\_exception('Lava');

CREATE OR REPLACE PROCEDURE exception\_handler(BRAND\_NAME VARCHAR)

RETURNS string

LANGUAGE SQL

EXECUTE AS CALLER

AS

DECLARE

brand\_count INT := 0;

no\_data\_found EXCEPTION (-20101, 'Brand does not exist');

BEGIN

SELECT COUNT(BRAND\_NAME) into :brand\_count

FROM my\_database.public.mobile\_discount

WHERE upper(BRAND\_NAME) = upper(:BRAND\_NAME);

let res string := (case when :brand\_count = 0 then '0'

else to\_varchar(:brand\_count)

end);

IF (brand\_count = '0') THEN

RAISE no\_data\_found;

END IF;

return res;

EXCEPTION

WHEN statement\_error THEN

return object\_construct('Error Type','statement\_error',

'SQLCODE',SQLCODE,

'SQLERRM',SQLERRM,

'SQLSTATE', SQLSTATE);

WHEN expression\_error THEN

return object\_construct('Error Type','expression\_error',

'SQLCODE',SQLCODE,

'SQLERRM',SQLERRM,

'SQLSTATE', SQLSTATE);

WHEN no\_data\_found THEN

return object\_construct('Error Type','User-Defined Exception',

'SQLCODE',SQLCODE,

'SQLERRM',SQLERRM,

'SQLSTATE', SQLSTATE);

END;

call exception\_handler('lava');

Tasks:-

CREATE OR ALTER TASK mobile

WAREHOUSE = 'TASKS',

SCHEDULE = 'USING CRON 55 5 \* \* \* Asia/Kolkata',

USER\_TASK\_TIMEOUT\_MS = 10,

SUSPEND\_TASK\_AFTER\_NUM\_FAILURES = 2,

COMMENT = 'SAMPLE\_TASK',

TASK\_AUTO\_RETRY\_ATTEMPTS = 2

AS

Create or replace view MY\_DATABASE.PUBLIC.MOBILE\_DISCOUNTS\_SAMPLE

as SELECT \* FROM MY\_DATABASE.PUBLIC.MOBILE\_DISCOUNT;

--------------------------------------------------------------------------------------------------------------------

view creation:-

CREATE OR REPLACE VIEW mobile\_discount AS (

SELECT

case

when product\_name like 'I Kall%' then 'I Kall'

when product\_name like 'CMF%' then 'Nothing'

else SPLIT\_PART(PRODUCT\_NAME, ' ', 1)

end AS brand\_name,

\*,

ROUND(((actual\_price - discount\_price) / actual\_price) \* 100) AS discount\_percent

FROM mobiles

);