-------------------------------------------------------------------------------XXXXXXXXXXX--------------------------------------------------------------------------------------------

--1st STEP

CREATE DATABASE MK\_DEMODATABASE

--2nd STEP

USE DATABASE MK\_DEMODATABASE

--3rd STEP

CREATE OR REPLACE TABLE MK\_INEURON\_CONSUMER\_COMPLAINTS

(

DATE\_RECEIVED STRING ,

PRODUCT\_NAME VARCHAR2(100) ,

SUB\_PRODUCT VARCHAR2(100) ,

ISSUE VARCHAR2(100),

SUB\_ISSUE VARCHAR2(100),

CONSUMER\_COMPLAINT\_NARRATIVE STRING,

Company\_Public\_Response STRING,

Company VARCHAR(100),

State\_Name CHAR(10),

Zip\_Code CHAR(10),

Tags VARCHAR(100),

Consumer\_Consent\_Provided VARCHAR(100),

Submitted\_via STRING,

Date\_Sent\_to\_Company STRING,

Company\_Response\_to\_Consumer VARCHAR(100),

Timely\_Response VARCHAR(100),

CONSUMER\_DISPUTED VARCHAR(100),

COMPLAINT\_ID NUMBER(12,0) NOT NULL PRIMARY KEY

);

CREATE OR REPLACE FILE FORMAT ConsumerComplaints

TYPE = CSV , SKIP\_HEADER = 1

--4TH STEP

--LOAD TABLE FROM LOCAL SYSTEM

--5TH STEP

--RUN SOME REAL LIFE QUERY

select \* from MK\_INEURON\_CONSUMER\_COMPLAINTS

SELECT \* FROM MK\_INEURON\_CONSUMER\_COMPLAINTS LIMIT 10000;

SELECT DISTINCT (PRODUCT\_NAME) from MK\_INEURON\_CONSUMER\_COMPLAINTS

--Consumer Loan

--Student loan

--Payday loan

--AS ABOVE 3 INTO SINGLE COLUMN AS LOAN TYPE

--AND REST OTHER AS "OTHER\_FINANC\_SERV".

SELECT \*,

CASE

WHEN PRODUCT\_NAME IN ('Consumer Loa','Student loan','Payday loan') THEN 'CSP\_LOAN'

ELSE 'OTHER\_FINANC\_SERV'

END AS LOAN\_TYPE

FROM MK\_INEURON\_CONSUMER\_COMPLAINTS;

--DISTENCT DATE

SELECT DISTINCT (DATE\_RECEIVED) FROM MK\_INEURON\_CONSUMER\_COMPLAINTS

--EXTRACT YEAR FROM DATE

SELECT SUBSTRING(DATE\_RECEIVED,7,12) FROM MK\_INEURON\_CONSUMER\_COMPLAINTS

SELECT DISTINCT SUBSTRING(DATE\_RECEIVED,7,12) FROM MK\_INEURON\_CONSUMER\_COMPLAINTS

--WE HAVE 2013-2015

*--1ST JAN 2013 -31 DEC 2013 -PPR12*

*--1ST JAN 2014 -31 DEC 2014 -PR12*

*--1ST JAN 2015 -31 DEC 2015 -R12*

*--NAME THE COLUMN AS 'TXN\_PERIOD'*

SELECT \*,

CASE

WHEN ((TO\_DATE(DATE\_RECEIVED, 'DD-MM-YYYY')) >= '2013-01-01' AND (TO\_DATE(DATE\_RECEIVED, 'DD-MM-YYYY')) <='2013-12-31')THEN 'PPR12\_PERIOD'

WHEN ((TO\_DATE(DATE\_RECEIVED, 'DD-MM-YYYY')) >= '2014-01-01' AND (TO\_DATE(DATE\_RECEIVED, 'DD-MM-YYYY')) <='2014-12-31') THEN 'PR12\_PERIOD'

WHEN ((TO\_DATE(DATE\_RECEIVED, 'DD-MM-YYYY')) >= '2015-01-01' AND (TO\_DATE(DATE\_RECEIVED, 'DD-MM-YYYY')) <='2015-12-31') THEN 'R12\_PERIOD'

ELSE 'INVALID\_RANGE'

END AS TXN\_PERIOD

FROM MK\_INEURON\_CONSUMER\_COMPLAINTS;

--------------------------------------------------------------------------XXXXXXXXXXXXXXXX----------------------------------------------------------------------------------------------------------------------

**--SQL DATE FUNCTION**

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

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

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

-- GET CURRENT DATE

SELECT CURRENT\_DATE;

SELECT CURRENT\_TIMESTAMP;

-- GET CURRENT DATE-- GET CURRENT TIME

SELECT CURRENT\_TIME;

-- CONVERT TIMEZONE

SELECT CONVERT\_TIMEZONE('UTC',CURRENT\_TIMESTAMP) AS UTC\_TIMEZONE;

-- CONVERT DATE TO SUBSEQUENT 4 MONTHS AHEAD

SELECT ADD\_MONTHS(CURRENT\_DATE,4) as DATE\_AFTER\_4\_MONTHS;

-- 3 MONTHS BACK DATE

SELECT TO\_CHAR(ADD\_MONTHS(CURRENT\_DATE,-3),'DD-MM-YYYY') as DATE\_BEFORE\_3\_MONTHS;

-- GET YR FROM DATE

SELECT DATE\_TRUNC('YEAR',CURRENT\_DATE) AS YR\_FROM\_DATE;

-- GET MTH FROM DATE

SELECT DATE\_TRUNC('MONTH',CURRENTA\_DATE) AS MTH\_FROM\_DATE;

-- GET DAY FROM DATE

SELECT DATE\_TRUNC('DAY',CURRENT\_DATE) AS DAY\_FROM\_DATE;

-- GET LAST DAY OF current MONTH

select last\_day(current\_date) as last\_day\_curr\_month;

-- GET LAST DAY OF PREVIOUS MONTH

SELECT LAST\_DAY(CURRENT\_DATE - INTERVAL '1 MONTH') AS LAST\_DAY\_PREV\_MNTH;

SELECT LAST\_DAY(CURRENT\_DATE - INTERVAL '2 MONTH') + INTERVAL '1 DAY' AS FIRST\_DAY;

SELECT QUARTER(CURRENT\_DATE) AS QTR;

SELECT EXTRACT(YEAR FROM CURRENT\_DATE) AS YR;

SELECT EXTRACT(MONTH FROM CURRENT\_DATE) AS MTH;

SELECT EXTRACT(DAY FROM CURRENT\_DATE) AS DAY;

select QUARTER(to\_date('2022-08-24'));

SELECT to\_date('08-23-2022','mm-dd-yyyy');

SELECT TO\_CHAR('08-23-2022','MM-DD-YYYY');

SELECT TO\_DATE('1993-08-17') AS DATE;

SELECT TO\_CHAR(TO\_DATE('1993-08-17'),'DD-MM-YYYY') AS DATE\_DD\_MM\_YYYY;

SELECT TO\_CHAR(TO\_DATE('1993-08-17'),'MM-YYYY') AS DATE\_MM\_YYYY;

SELECT TO\_CHAR(TO\_DATE('1993-08-17'),'MON-YYYY') AS DATE\_MON\_YYYY;

SELECT TO\_CHAR(TO\_DATE('1993-08-17'),'MON-YY') AS DATE\_MON\_YY;

SELECT TO\_CHAR(TO\_DATE('1993-08-17'),'DY') AS DATE\_DAY;

SELECT DAYNAME ('1993-08-23');

SELECT TO\_CHAR(TO\_DATE('1993-08-17'),'YYYY-DD') AS DATE;

SELECT TO\_CHAR(TO\_DATE('1993-08-17'),'DD-MM') AS DATE;

select MONTH(CURRENT\_DATE);

SELECT EXTRACT(MONTH FROM CURRENT\_DATE) AS MTH;

SELECT ADD\_MONTHS(CURRENT\_DATE,-3) AS DATE\_3\_MNTHS\_BACK;

SELECT ADD\_MONTHS(CURRENT\_DATE,5) AS DATE\_5\_MNTHS\_AHEAD;

select datediff('day', '2022-06-01',CURRENT\_DATE);

select datediff('day', '2022-07-23','2023-07-19');

select datediff('MONTH', '2021-06-01',CURRENT\_DATE);

select datediff('YEAR', '2014-06-01',CURRENT\_DATE);

select dateadd('day',-23,'2022-06-01');

select dateadd('month',-2,'2022-06-01');

select dateadd('year',-5,'2022-06-01');

select WEEK(CURRENT\_DATE); -- *FROM 1ST JAN 2022 HOW MNAY EEKS HAVE SURPASSED*

select MONTH(CURRENT\_DATE); *-- -- FROM 1ST JAN 2022 HOW MNAY MONTHS HAVE SURPASSED*

select YEAR(AJSHBJCASbcsasb,); ----

select datediff('MONTH', '2022-06-01',CURRENT\_DATE);

select datediff('YEAR', '2014-06-01',CURRENT\_DATE);

SELECT DATE\_TRUNC('DAY',CURRENT\_DATE) AS DAY\_FROM\_DATE;

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

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

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

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

-------------------------------------------------------------------------------XXXXXXXXXXXXX----------------------------------------------------------------------------------------------------------------

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

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

**--WINDOWS FUNCTION**

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

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

CREATE OR REPLACE TABLE EMPLOYEE

(

EMPID INTEGER NOT NULL PRIMARY KEY,

EMP\_NAME VARCHAR2(20),

JOB\_ROLE STRING,

SALARY NUMBER(10,2)

);

INSERT INTO EMPLOYEE

VALUES('101','ANAND JHA','Analyst',50000),(102,'ALex', 'Data Enginner',60000),(103,'Ravi', 'Data Scientist',48000),

(104,'Peter', 'Analyst',98000),(105,'Pulkit', 'Data Scientist',72000),(106,'Robert','Analyst',100000),

(107,'Rishabh','Data Engineer',67000),(108,'Subhash','Manager',148000),(109,'Michaeal','Manager',213000),

(110,'Dhruv','Data Scientist',89000),(111,'Amit Sharma','Analyst',72000);

SELECT \* FROM EMPLOYEE;

update employee

set job\_role='Data Engineer'

where empid=102;

update employee

set salary= 50000

where empid=104;

*-- SYNTAX : window\_function\_name(<exprsn>) OVER (<partition\_by\_clause> <order\_clause>)*

--- display total salary based on job profile

SELECT JOB\_ROLE,SUM(SALARY) FROM EMPLOYEE

GROUP BY 1

----ORDER BY 2;

---ADD FUNCTION ROW\_NUMBER---

SELECT \*,

ROW\_NUMBER() OVER(ORDER BY JOB\_ROLE) AS ROW\_NUM

FROM EMPLOYEE;

-- display total salary along with all the records ()every row value

SELECT \* ,

SUM(SALARY) OVER() AS TOT\_SALARY

FROM EMPLOYEE;

-- display the total salary per job category for all the rows

SELECT \*,

MAX(SALARY) OVER(PARTITION BY JOB\_ROLE) AS MAX\_JOB\_SAL

FROM EMPLOYEE;

select \*,

max(salary) over(partition by JOB\_ROLE) as MAX\_SAL ,

min(salary) over(partition by JOB\_ROLE) as MIN\_SAL,

SUM(salary) over(partition by JOB\_ROLE) as TOT\_SAL

from Employee;

--ARRANGING ROWS WITHIN EACH PARTITION BASED ON SALARY IN DESC ORDER

select \*,

max(salary) over(partition by JOB\_ROLE ORDER BY SALARY DESC) as MAX\_SAL ,

min(salary) over(partition by JOB\_ROLE ORDER BY SALARY DESC) as MIN\_SAL,

SUM(salary) over(partition by JOB\_ROLE ORDER BY SALARY DESC) as TOT\_SAL

from Employee;

-- ROW\_NUMBER() It assigns a unique sequential number to each row of the table ...

SELECT \* FROM

(

SELECT \*, ROW\_NUMBER() OVER(PARTITION BY JOB\_ROLE ORDER BY SALARY DESC) AS PART\_ROW\_NUM

FROM EMPLOYEE

)

WHERE PART\_ROW\_NUM <=2;

*/\* The RANK() window function, as the name suggests, ranks the rows within their partition based on the given condition.*

*In the case of ROW\_NUMBER(), we have a sequential number.*

*On the other hand, in the case of RANK(), we have the same rank for rows with the same value.*

*But there is a problem here. Although rows with the same value are assigned the same rank, the subsequent rank skips the missing rank.*

*This wouldn’t give us the desired results if we had to return “top N distinct” values from a table.*

*Therefore we have a different function to resolve this issue. \*/*

SELECT \*,

ROW\_NUMBER() OVER(PARTITION BY JOB\_ROLE ORDER BY SALARY DESC) AS ROW\_NUM ,

RANK() OVER(PARTITION BY JOB\_ROLE ORDER BY SALARY DESC) AS RANK\_ROW

FROM EMPLOYEE;

*/\* The DENSE\_RANK() function is similar to the RANK() except for one difference, it doesn’t skip any ranks when ranking rows*

*Here, all the ranks are distinct and sequentially increasing within each partition.*

*As compared to the RANK() function, it has not skipped any rank within a partition. \*/*

SELECT \* FROM

(

SELECT \*,

ROW\_NUMBER() OVER(PARTITION BY JOB\_ROLE ORDER BY SALARY) AS ROW\_NUM ,

RANK() OVER(PARTITION BY JOB\_ROLE ORDER BY SALARY) AS RANK\_ROW,

DENSE\_RANK() OVER(PARTITION BY JOB\_ROLE ORDER BY SALARY DESC) AS DENSE\_RANK\_ROW

FROM EMPLOYEE

)

WHERE DENSE\_RANK\_ROW <=2;

-------------------------------------------------------------------------------XXXXXXXXXXXXXX---------------------------------------------------------------------------------

CREATE OR REPLACE TABLE CONSUM\_COMPLAINTS\_AWS LIKE MK\_INEURON\_CONSUMER\_COMPLAINTS;

SHOW COLUMNS IN CONSUM\_COMPLAINTS\_AWS;

----------------------------------------------------------------------------------XXXXXXXXXXXXXXXXXX--------------------------------------------------------------------------------------------------

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

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

**--CONTINUOUS DATA LOADING**

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

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

CREATE DATABASE DATA\_PIPELINING;

USE DATA\_PIPELINING;

CREATE TABLE patients

(

PATIENT\_ID INT,

FIRST\_NAME VARCHAR(100),

CITY VARCHAR(50),

REGISTRATION\_YEAR INT

);

---CREATE STORAGE INTEGRATION

CREATE OR REPLACE STORAGE integration aj\_s3\_int

TYPE = EXTERNAL\_STAGE

STORAGE\_PROVIDER = S3

ENABLED = TRUE

STORAGE\_AWS\_ROLE\_ARN ='arn:aws:iam::441615131317:role/snowpipe-data-injection'

STORAGE\_ALLOWED\_LOCATIONS =('s3://patientsdatasnowpipe/');

desc integration aj\_s3\_int;

--create the stage

create or replace stage AJ\_PATIENTS\_SNOW\_STAGE

url = 's3://patientsdatasnowpipe/' -- name of the bucket which you have created

*--credentials = (aws\_key\_id = 'AKIAWNUSQEK24DDN3WP3' aws\_secret\_key = 'UsT6YqwfU0ag5ydUUP1gGD7UPQ6sUA335M5zueP+')*

file\_format = CSV

storage\_integration =aj\_s3\_int;

SHOW STAGES;

*--CREATE SNOWPIPE THAT RECOGNISES CSV THAT ARE INGESTED FROM EXTERNAL STAGE AND COPIES THE DATA INTO PATIENTS 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 AJ\_PATIENTS\_SNOWPIPE AUTO\_INGEST = TRUE AS

COPY INTO "DATA\_PIPELINING"."PUBLIC"."PATIENTS"

FROM @AJ\_PATIENTS\_SNOW\_STAGE

FILE\_FORMAT = (type= 'CSV');

SHOW PIPES;

SELECT \* FROM PATIENTS;

select count(\*) from patients;

alter pipe PATIENTS\_SNOWPIPE refresh;

CREATE OR REPLACE TABLE HEALTHCARE(

Patientid VARCHAR(15),

gender CHAR(8),

age VARCHAR(5) ,

hypertension CHAR(20),

heart\_disease CHAR(20),

ever\_married CHAR(30),

work\_type VARCHAR(60),

Residence\_type CHAR(30) ,

avg\_glucose\_level VARCHAR(20),

bmi VARCHAR(20) ,

smoking\_status VARCHAR(20),

stroke CHAR(20)

);

--CREATE OR REPLACE STORAGE integration s3\_int

CREATE OR REPLACE STORAGE integration s3\_int

TYPE = EXTERNAL\_STAGE

STORAGE\_PROVIDER = S3

ENABLED = TRUE

STORAGE\_AWS\_ROLE\_ARN ='arn:aws:iam::516104601311:role/snowpiperole'

STORAGE\_ALLOWED\_LOCATIONS =('s3://awshealthcarebucket');

desc integration s3\_int;

CREATE OR REPLACE STAGE HEALTHCARE\_SP

URL ='s3://awshealthcarebucket'

credentials=(aws\_key\_id='AKIAXQKR3H3PSG72XFMK'aws\_secret\_key='eKL6a6FjlQHic4s8Ne712Aelzg2ou4j6tNsVvFq5')

file\_format=CSV;

--storage\_integration =s3\_int;

LIST @HEALTHCARE\_SP;

SHOW STAGES;

*--CREATE SNOWPIPE THAT RECOGNISES CSV THAT ARE INGESTED FROM EXTERNAL STAGE AND COPIES THE DATA INTO PATIENTS 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 HEALTHCARE\_SNOWPIPE AUTO\_INGEST = TRUE AS

COPY INTO "DATA\_PIPELINING"."PUBLIC"."HEALTHCARE"

FROM @HEALTHCARE\_SP

FILE\_FORMAT = CSV;

SHOW PIPES;

SELECT count(\*) FROM HEALTHCARE;

select \* from HEALTHCARE;

alter pipe HEALTHCARE\_SNOWPIPE refresh;

SELECT SYSTEM$PIPE\_STATUS('HEALTHCARE\_SNOWPIPE');