```sql
--Create Database and Schema

CREATE DATABASE Sales_Project;

USE Sales_Project;

--Create and Configure Warehouse

CREATE OR REPLACE WAREHOUSE Sales

WITH

   WAREHOUSE_SIZE = 'X-Small'

   AUTO_SUSPEND = 300

   AUTO_RESUME = TRUE

   MIN_CLUSTER_COUNT = 1

   MAX_CLUSTER_COUNT = 3;

CREATE OR REPLACE SCHEMA Sale;

---Adjust warehouse size due to data volume (~1 GB)

ALTER WAREHOUSE Sales

SET WAREHOUSE_SIZE = 'SMALL';

USE DATABASE Sales_Project;

USE SCHEMA Sale;

--Create Internal Stage

CREATE OR REPLACE STAGE int_stg

COMMENT = 'Internal Stage for Data Loading';

-- Load Data from Internal Stage into the Staging Table

COPY INTO raw_sales_data1

FROM @int_stg/sale_1.parquet

FILE_FORMAT = (TYPE = PARQUET)

ON_ERROR = 'CONTINUE'

FORCE = TRUE;

--Create Raw Staging Table

CREATE OR REPLACE TABLE raw_sales_data1(

   raw_data VARIANT

)
```

```
COMMENT = 'Raw sales data staging table before transformations';

SELECT COUNT(*) FROM raw_sales_data1;

SELECT * FROM raw_sales_data1;--count of rows

---Creating a table to load data from raw_data

CREATE OR REPLACE TABLE staging_sales AS(

SELECT

 raw_data:brand::VARCHAR AS brand,

 raw_data:category::VARCHAR AS category,

 raw_data:city::VARCHAR AS city,

 raw_data:customer_age_group::VARCHAR AS customer_age_group,

 raw_data:customer_email::VARCHAR AS customer_email,

 raw_data:customer_id::INT AS customer_id,

 raw_data:customer_name::VARCHAR AS customer_name,

 raw_data:customer_segment::VARCHAR AS customer_segment,

 raw_data:delivery_status::VARCHAR AS delivery_status,

 raw_data:discount::NUMBER(10,2) AS discount,

 raw_data:location_id::INT AS location_id,

 TO_DATE(raw_data:$1:order_date::VARCHAR, 'DD-MM-YYYY') AS order_date,

 raw_data:order_id:: INT AS order_id,

 raw_data:order_time:: TIME AS order_time,

 raw_data:payment_method::VARCHAR AS payment_method,

 raw_data:postal_code::VARCHAR AS postal_code,

 raw_data:product_id::INT AS product_id,

 raw_data:product_name::VARCHAR AS product_name,

 raw_data:province::VARCHAR AS province,

 raw_data:quantity::INT AS quantity,

 raw_data:region::VARCHAR AS region,

 TO_DATE(raw_data:$1:shipping_date::VARCHAR, 'DD-MM-YYYY') AS shipping_date,

 raw_data:tax::NUMBER(10,2) AS tax,

 raw_data:total_amount::NUMBER(10,2) AS total_amount,
```

```sql
  raw_data:unit_price::NUMBER(10,2) AS unit_price
FROM
   raw_sales_data1);
SELECT * FROM STAGING_SALES;
--Creating storage integration to auto ingest from s3 to snowflake
CREATE OR REPLACE STORAGE INTEGRATION aws_s3_integration
TYPE = EXTERNAL_STAGE
STORAGE_PROVIDER = 'S3'
ENABLED = TRUE
STORAGE_AWS_ROLE_ARN ='arn:aws:iam::759489707541:role/sqlpractice'
STORAGE_ALLOWED_LOCATIONS = ('s3://sqlpractice25/');
DESCRIBE INTEGRATION aws_s3_integration;
---Load Parquet Data into Raw Table
CREATE OR REPLACE STAGE ext_stg
URL= 's3://sqlpractice25/'
STORAGE_INTEGRATION = aws_s3_integration
FILE_FORMAT = (TYPE = PARQUET);
DESCRIBE INTEGRATION aws_s3_integration;
---Using tasks for auto-data loading using CRON
CREATE OR REPLACE TASK aws_data_load_task
   WAREHOUSE = 'SALES'
   SCHEDULE = 'USING CRON 0 1 * * * Asia/Kolkata'
AS
COPY INTO raw_sales_data1
FROM @ext_stg/
FILE_FORMAT = (TYPE = PARQUET)
ON_ERROR = 'CONTINUE'
PURGE = TRUE;
ALTER TASK aws_data_load_task SUSPEND;
ALTER TASK aws_data_load_task
```

```
SET SCHEDULE = 'USING CRON 45 9 * * * Asia/Kolkata';


ALTER TASK aws_data_load_task RESUME;

LIST @ext_stg;

LIST @int_stg;

SELECT COUNT(*) FROM raw_sales_data1;

SELECT

   raw:data:

   raw_data:order_id::INT                AS order_id,

   raw_data:order_date::DATE             AS order_date,

   raw_data:order_time::TIME             AS order_time,

   raw_data:shipping_date::DATE          AS shipping_date,

   raw_data:delivery_status::VARCHAR       AS delivery_status,

   raw_data:payment_method::VARCHAR       AS payment_method,

   raw_data:product_id::INT              AS product_id,

   raw_data:product_name::VARCHAR         AS product_name,

   raw_data:category::VARCHAR            AS category,

   raw_data:brand::VARCHAR               AS brand,

   raw_data:unit_price::NUMBER(10, 2)     AS unit_price,

   raw_data:quantity::INT                AS quantity,

   raw_data:discount::NUMBER(5, 2)        AS discount_rate,

   raw_data:tax::NUMERIC(10, 2)          AS tax_amount,

   raw_data:total_amount::NUMERIC(10, 2)   AS total_amount,

   raw_data:customer_id::INT             AS customer_id,

   raw_data:customer_name::VARCHAR        AS customer_name,

   raw_data:customer_email::VARCHAR       AS customer_email,

   raw_data:customer_segment::VARCHAR      AS customer_segment,

   raw_data:customer_age_group::VARCHAR    AS customer_age_group,

   raw_data:location_id::INT             AS location_id,

   raw_data:city::VARCHAR                AS city,
   raw_data:province::VARCHAR            AS province,
   raw_data:region::VARCHAR              AS region,
   raw_data:postal_code::VARCHAR         AS postal_code
FROM
   STAGING_SALES;
LIST @int_stg;
```

---Creating a temporary table to load csv file

```
CREATE OR REPLACE TABLE csv_staging_temp (
  PRODUCT_ID INT,
  PRODUCT_NAME VARCHAR,
  CATEGORY VARCHAR,
  BRAND VARCHAR,
  ORDER_ID INT,
  ORDER_DATE VARCHAR,
  ORDER_TIME VARCHAR,
  SHIPPING_DATE VARCHAR,
  DELIVERY_STATUS VARCHAR,
  CUSTOMER_ID INT,
  CUSTOMER_NAME VARCHAR,
  CUSTOMER_EMAIL VARCHAR,
  CUSTOMER_SEGMENT VARCHAR,
  CUSTOMER_AGE_GROUP VARCHAR,
  LOCATION_ID INT,
  CITY VARCHAR,
  PROVINCE VARCHAR,
  REGION VARCHAR,
  POSTAL_CODE VARCHAR,
  QUANTITY INT,
  UNIT_PRICE NUMBER(10, 2),
  DISCOUNT NUMBER(10, 2),
  TAX NUMBER(10, 2),

  TOTAL_AMOUNT NUMBER(10, 2),

  PAYMENT_METHOD VARCHAR

);
```

--Creating file format for csv file

```
CREATE OR REPLACE FILE FORMAT my_csv_format

TYPE = CSV

SKIP_HEADER = 1

TRIM_SPACE = TRUE

FIELD_OPTIONALLY_ENCLOSED_BY = ' " '

EMPTY_FIELD_AS_NULL = TRUE

DATE_FORMAT = 'DD-MM-YYYY'

TIME_FORMAT = 'HH24:MI:SS';
```

---Load CSV Data into Temp Table

```
COPY INTO csv_staging_temp

FROM @int_stg/sales5.csv.gz

FILE_FORMAT = my_csv_format

ON_ERROR = CONTINUE;


SELECT COUNT(*) FROM csv_staging_temp;
```

---MERGE Data from CSV Temp into Staging Table

```
MERGE INTO staging_sales s  --- Target table

USING csv_staging_temp c   --- Source table

ON s.ORDER_ID = c.ORDER_ID

WHEN MATCHED THEN

   UPDATE SET

      s.PRODUCT_ID = c.PRODUCT_ID,

      s.PRODUCT_NAME = c.PRODUCT_NAME,

      s.CATEGORY = c.CATEGORY,
      s.BRAND = c.BRAND,
      s.ORDER_DATE = TO_DATE(c.ORDER_DATE, 'DD-MM-YYYY'),
      s.ORDER_TIME = TO_TIME(c.ORDER_TIME, 'HH24:MI:SS'),
      s.SHIPPING_DATE = TO_DATE(c.SHIPPING_DATE, 'DD-MM-YYYY'),
      s.DELIVERY_STATUS = c.DELIVERY_STATUS,
      s.CUSTOMER_ID = c.CUSTOMER_ID,
      s.CUSTOMER_NAME = c.CUSTOMER_NAME,
      s.CUSTOMER_EMAIL = c.CUSTOMER_EMAIL,
      s.CUSTOMER_SEGMENT = c.CUSTOMER_SEGMENT,
      s.CUSTOMER_AGE_GROUP = c.CUSTOMER_AGE_GROUP,
      s.LOCATION_ID = c.LOCATION_ID,
      s.CITY = c.CITY,
      s.PROVINCE = c.PROVINCE,
      s.REGION = c.REGION,
      s.POSTAL_CODE = c.POSTAL_CODE,
      s.QUANTITY = c.QUANTITY,
      s.UNIT_PRICE = c.UNIT_PRICE,
      s.DISCOUNT = c.DISCOUNT,
      s.TAX = c.TAX,
      s.TOTAL_AMOUNT = c.TOTAL_AMOUNT,
      s.PAYMENT_METHOD = c.PAYMENT_METHOD
WHEN NOT MATCHED THEN
   INSERT (
      PRODUCT_ID, PRODUCT_NAME, CATEGORY, BRAND, ORDER_ID, ORDER_DATE, ORDER_TIME,
      SHIPPING_DATE, DELIVERY_STATUS, CUSTOMER_ID, CUSTOMER_NAME, CUSTOMER_EMAIL,
      CUSTOMER_SEGMENT, CUSTOMER_AGE_GROUP, LOCATION_ID, CITY, PROVINCE, REGION,
      POSTAL_CODE, QUANTITY, UNIT_PRICE, DISCOUNT, TAX, TOTAL_AMOUNT, PAYMENT_METHOD
      )
   VALUES (
      c.PRODUCT_ID, c.PRODUCT_NAME, c.CATEGORY, c.BRAND, c.ORDER_ID,
      TO_DATE(c.ORDER_DATE, 'DD-MM-YYYY'), TO_TIME(c.ORDER_TIME, 'HH24:MI:SS'), TO_DATE(c.SHIPPING_DATE, 'DD-MM-YYYY'),c.DELIVERY_STATUS, c.CUSTOMER_ID, c.CUSTOMER_NAME, c.CUSTOMER_EMAIL,

      c.CUSTOMER_SEGMENT, c.CUSTOMER_AGE_GROUP, c.LOCATION_ID, c.CITY, c.PROVINCE, c.REGION,

      c.POSTAL_CODE, c.QUANTITY, c.UNIT_PRICE, c.DISCOUNT, c.TAX, c.TOTAL_AMOUNT, c.PAYMENT_METHOD

   );


SELECT COUNT(*) FROM STAGING_SALES;
```

---dimensional model (Star Schema) creation tables define the final, corrected **Star Schema architecture**, clear **Fact/Dimension** separation for optimal analytical performance

---Create FACT_SALES Table

```
CREATE OR REPLACE TABLE fact_sales(

  PRODUCT_ID INT,

  ORDER_ID INT,

  CUSTOMER_ID INT,

  LOCATION_ID INT,

  DISCOUNT NUMBER(10,2),

  TAX NUMBER(10,2),

  Total_amount NUMBER(10,2)

);
```

---Create DIM_ORDER Table

```
CREATE OR REPLACE TABLE dim_order(

  ORDER_ID INT,

  ORDER_DATE VARCHAR,

  ORDER_TIME VARCHAR,

  SHIPPING_DATE VARCHAR,

  DELIVERY_STATUS VARCHAR,

  PAYMENT_METHOD VARCHAR

);
```

---Create DIM_PRODUCT Table

```
CREATE OR REPLACE TABLE dim_product(
    PRODUCT_ID INT,
    PRODUCT_NAME VARCHAR,
    CATEGORY VARCHAR,
    BRAND VARCHAR,
    UNIT_PRICE NUMBER(10,2),
    QUANTITY INT
);
```

---Create DIM_CUSTOMER Table

```
CREATE OR REPLACE TABLE dim_customer(
    CUSTOMER_ID INT,
    CUSTOMER_NAME VARCHAR,
    CUSTOMER_EMAIL VARCHAR,
    CUSTOMER_SEGMENT VARCHAR,
    CUSTOMER_AGE_GROUP INT
);
```

---Create DIM_LOCATION Table

```
CREATE OR REPLACE TABLE dim_location(
    LOCATION_ID INT,
    CITY VARCHAR,
    PROVIENCE VARCHAR,
    REGION VARCHAR,
    POSTAL_CODE VARCHAR
);
```