Data Analysis with Apache Hive on Telecom Data

1. Data Loading (Beginner)

```
gcloud compute scp CustomerDemographics.csv
jainishpatelwork@cluster-8ac7-m:/home/jainishpatelwork

gcloud compute scp Telecom_customer_churn_data.csv
jainishpatelwork@cluster-8ac7-m:/home/jainishpatelwork
```

a. Download the dataset and load it into a Hive table.

```
CREATE TABLE IF NOT EXISTS tele_customer_churn(
    customerID STRING,
    gender STRING,
    SeniorCitizen INT,
    Partner STRING,
    Dependents STRING,
    tenure INT,
    PhoneService STRING,
    MultipleLines STRING,
    InternetService STRING,
    OnlineSecurity STRING,
    OnlineBackup STRING,
    DeviceProtection STRING,
    TechSupport STRING,
    StreamingTV STRING,
    StreamingMovies STRING,
    Contract STRING,
    PaperlessBilling STRING,
    PaymentMethod STRING,
    MonthlyCharges INT,
    TotalCharges INT,
    Churn STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
tblproperties("skip.header.line.count"="1") ;

load data local inpath
'file:///home/jainishpatelwork/Telecom_customer_churn_data.csv
' into table tele_customer_churn;
```

b. Write a query to display the top 10 rows of the table.

```
select * from tele_customer_churn limit 10;

select * from cust_demo limit 10;
```

2. Data Exploration (Beginner)

a. Write a HiveQL query to find the total number of customers in the Dataset.

```
select COUNT(DISTINCT customerID) from tele_customer_churn;
```

b. Write a HiveQL query to find the number of customers who have Churned.

```
select COUNT(DISTINCT customerID) from tele_customer_churn
where Churn = 'Yes';
```

c. Analyze the distribution of customers based on gender and SeniorCitizen status.

```
select gender, SeniorCitizen, COUNT(*) from
tele_customer_churn group by gender, SeniorCitizen;
```

d. Determine the total charge to the company due to churned Customers.

```
select SUM(TotalCharges) as TotalCharges from
tele_customer_churn where Churn = 'Yes';
```

3. Data Analysis (Intermediate)

a. Write a HiveQL query to find the number of customers who have churned, grouped by their Contract type.

```
SELECT
Contract,
COUNT(*) as num_of_cust
FROM tele_customer_churn
WHERE Churn = 'Yes'
```

```
GROUP BY Contract;
```

b. Write a HiveQL query to find the average MonthlyCharges for customers who have churned vs those who have not.

```
SELECT
Churn,
AVG(MonthlyCharges) as avg_charge
FROM tele_customer_churn
GROUP BY Churn;
```

c. Determine the maximum, minimum, and average tenure of the customers.

```
SELECT
MAX(tenure) as max_tenure,
MIN(tenure) as min_tenure,
AVG(tenure) as avg_tenure
FROM tele_customer_churn;
```

d. Find out which PaymentMethod is most popular among customers.

```
SELECT
PaymentMethod,
COUNT(*) as Total_Count
FROM tele_customer_churn
GROUP BY PaymentMethod
ORDER BY COUNT(*) DESC;
```

e. Analyze the relationship between PaperlessBilling and churn rate.

```
SELECT
PaperlessBilling,
COUNT(*) as total_cust,
ROUND(AVG(CASE WHEN Churn = 'Yes' THEN 1 ELSE 0 END),2)*100.0
as churn_rate
FROM tele_customer_churn
GROUP BY PaperlessBilling;
```

4. Partitioning (Intermediate)

a. Create a partitioned table by Contract and load the data from the

original table.

```
SET hive.exec.dynamic.partition = true;
SET hive.exec.dynamic.partition.mode = nonstrict;

CREATE TABLE IF NOT EXISTS
tele_customer_churn_part_by_contract(
    customerID STRING,
    gender STRING,
    SeniorCitizen INT,
    Partner STRING,
    Dependents STRING,
    tenure INT,
    PhoneService STRING,
    MultipleLines STRING,
    InternetService STRING,
    OnlineSecurity STRING,
    OnlineBackup STRING,
    DeviceProtection STRING,
    TechSupport STRING,
    StreamingTV STRING,
    StreamingMovies STRING,
    PaperlessBilling STRING,
    PaymentMethod STRING,
    MonthlyCharges INT,
    TotalCharges INT,
    Churn STRING
)
PARTITIONED BY (Contract STRING);


INSERT OVERWRITE TABLE tele_customer_churn_part_by_contract
PARTITION (Contract)
SELECT
    customerID, gender, SeniorCitizen, Partner, Dependents,
    tenure, PhoneService, MultipleLines, InternetService,
    OnlineSecurity, OnlineBackup, DeviceProtection,
TechSupport,
    StreamingTV, StreamingMovies, PaperlessBilling,
PaymentMethod,
    MonthlyCharges, TotalCharges, Churn,
    Contract
FROM tele_customer_churn;
```

b. Write a HiveQL query to find the number of customers who have churned in each Contract type using the partitioned table.

```
SELECT
Contract,
COUNT(*) as total_cust
FROM tele_customer_churn_part_by_contract
WHERE Churn = 'Yes'
GROUP BY Contract;
```

c. Find the average MonthlyCharges for each type of Contract using the partitioned table.

```
SELECT
Contract,
AVG(MonthlyCharges) as AVG_MonthlyCharges
FROM tele_customer_churn_part_by_contract
GROUP BY Contract;
```

d. Determine the maximum tenure in each Contract type partition.

```
SELECT
Contract,
MAX(tenure) as tenure_max
FROM tele_customer_churn_part_by_contract
GROUP BY Contract;
```

5. Bucketing (Advanced)

a. Create a bucketed table by tenure into 6 buckets.

```
CREATE TABLE IF NOT EXISTS tele_customer_churn_bucket_tenure(
    customerID STRING,
    gender STRING,
    SeniorCitizen INT,
    Partner STRING,
    Dependents STRING,
    tenure INT,
    PhoneService STRING,
    MultipleLines STRING,
    InternetService STRING,
    OnlineSecurity STRING,
    OnlineBackup STRING,
    DeviceProtection STRING,
```

```
    TechSupport STRING,
    StreamingTV STRING,
    StreamingMovies STRING,
    Contract STRING,
    PaperlessBilling STRING,
    PaymentMethod STRING,
    MonthlyCharges INT,
    TotalCharges INT,
    Churn STRING
)
CLUSTERED BY (tenure)
sorted by(customerID)
into 6 buckets;
```

b. Load the data from the original table into the bucketed table.

```
insert overwrite table tele_customer_churn_bucket_tenure
select * from tele_customer_churn;
```

c. Write a HiveQL query to find the average MonthlyCharges for customers in each bucket.

```
SELECT
    abs(hash(tenure) % 6) AS Bucket_Index,
    COUNT(*) AS Customer_Count,
    AVG(MonthlyCharges) AS Avg_Monthly_Charges
FROM tele_customer_churn_bucket_tenure
GROUP BY abs(hash(tenure) % 6);
```

d. Find the highest TotalCharges in each tenure bucket.

```
SELECT
    (hash(tenure) % 6) AS bucket_number,
    COUNT(*) as customer_count,
    MAX(TotalCharges) as total_total_charges
FROM tele_customer_churn_bucket_tenure
GROUP BY (hash(tenure) % 6);
```

6. Performance Optimization with Joins (Advanced)

Assume another dataset, CustomerDemographics.csv, that contains the details of the demographic data of each customer.

a. Load the demographics dataset into another Hive table.

```
CREATE TABLE IF NOT EXISTS cust_demo(
```

```
    CustomerID STRING,
    City STRING,
    Lat FLOAT,
    Long FLOAT,
    Country STRING,
    iso2 STRING,
    State STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
tblproperties("skip.header.line.count"="1") ;

load data local inpath
'file:///home/jainishpatelwork/CustomerDemographics.csv' into
table cust_demo;
```

b. Write HiveQL queries to join the customer churn table and the demographics table on customerID using different types of joins - common join, map join, bucket map join, and sorted merge bucket join.

```
SELECT
   d.State, COUNT(*) total_churn
FROM tele_customer_churn c
JOIN cust_demo d ON c.customerID = d.CustomerID
GROUP BY d.State;



SELECT
   d.State, COUNT(*) total_churn
FROM tele_customer_churn c
LEFT JOIN cust_demo d ON c.customerID = d.CustomerID
GROUP BY d.State;

-- Enable Map Join
SET hive.auto.convert.join=true;
SET hive.mapjoin.smalltable.filesize=25000000;
SELECT
   d.State, COUNT(*) total_churn
FROM tele_customer_churn c
JOIN cust_demo d ON c.customerID = d.CustomerID
GROUP BY d.State;

SET hive.optimize.bucketmapjoin = true;
```

```
SELECT
d.State,
c.tenure,
AVG(c.MonthlyCharges) as AVG_MonthlyCharges
FROM tele_customer_churn_bucket_tenure as c
JOIN cust_demo d ON c.customerID = d.CustomerID
GROUP BY d.State,tenure;
```

c. Observe and document the performance of each join type.

7. Advanced Analysis (Expert)

a. Find the distribution of PaymentMethod among churned customers.

```
SELECT
PaymentMethod,
COUNT(*) as total_cust
FROM tele_customer_churn
WHERE Churn = 'Yes'
GROUP BY PaymentMethod;
```

b. Calculate the churn rate (percentage of customers who left) for each InternetService category.

```
SELECT
InternetService,
COUNT(*) as total_cust,
ROUND(AVG(CASE WHEN Churn = 'Yes' THEN 1 ELSE 0 END),2)*100.0
as churn_rate
FROM tele_customer_churn
GROUP BY InternetService;
```

c. Find the number of customers who have no dependents and have churned, grouped by Contract type.

```
SELECT
Contract,
COUNT(*) as total_cust
FROM tele_customer_churn
WHERE Dependents = 'No' and Churn = 'Yes'
GROUP BY Contract;
```

d. Find the top 5 tenure lengths that have the highest churn rates.

```
SELECT
tenure,
COUNT(*) as total_cust
FROM tele_customer_churn
WHERE Churn = 'Yes'
GROUP BY tenure
ORDER BY COUNT(*) DESC LIMIT 5;
```

e. Calculate the average MonthlyCharges for customers who have PhoneService and have churned, grouped by Contract type.

```
SELECT
Contract,
AVG(MonthlyCharges) as avg_monthly_charges
FROM tele_customer_churn
WHERE Churn = 'Yes' and PhoneService = 'Yes'
GROUP BY Contract;
```

f. Identify which InternetService type is most associated with churned customers.

```
SELECT
InternetService,
COUNT(*) as total_cust
FROM tele_customer_churn
WHERE Churn = 'Yes'
GROUP BY InternetService
ORDER BY COUNT(*) DESC;
```

g. Determine if customers with a partner have a lower churn rate compared to those without.

```
SELECT
Partner,
COUNT(*) as total_cust
FROM tele_customer_churn
WHERE Churn = 'Yes'
GROUP BY Partner
ORDER BY COUNT(*) DESC;
```

h. Analyze the relationship between MultipleLines and churn rate.

```
SELECT
MultipleLines,
ROUND(AVG(CASE WHEN Churn = 'Yes' THEN 1 ELSE 0 END),2)*100.0
as churn_rate
FROM tele_customer_churn
GROUP BY MultipleLines
ORDER BY COUNT(*) DESC;
```