Week 01 Descriptive Analysis

```
show tables;

SELECT DATABASE();
USE `capstone dataset`;

SHOW TABLES;
DESCRIBE `hospitalisationdetails`;
DESCRIBE `medicalexaminations`;
DESCRIBE `names`;


SHOW GRANTS FOR CURRENT_USER;

Select * from hospitalisationdetails;

Describe hospitalisationdetails;
SELECT `Customer ID` FROM hospitalisationdetails;
```

-- Colate the files together to form a master table aka 'MasterData'

```
CREATE TABLE MasterData AS
SELECT
   H.`Customer ID`,
   H.year,
   H.month,
   H.date,
   H.children,
   H.charges,
   H.`Hospital Tier`,
   H.`City Tier`,
   H.`State ID`,
   M.BMI,
   M.HBA1C,
   M.`Heart Issues`,
   M.`Any Transplants`,
   M.`Cancer History`,
   M.`NumberOfMajorSurgeries`,
   M.smoker,
   N.Name
FROM `hospitalisationdetails` H
JOIN `medicalexaminations` M ON H.`Customer ID` = M.`Customer ID`
JOIN `names` N ON H.`Customer ID` = N.`Customer ID`;
```

```
-- Checking to see if there are any null or missing values from our dataset
-- Diving deep into each column to check the same using COUNT function.

    select * from MasterData;
    SELECT
       COUNT(*) - COUNT(`Customer ID`) AS missing_CustomerID,
       COUNT(*) - COUNT(`year`) AS missing_year,
       COUNT(*) - COUNT(`month`) AS missing_month,
       COUNT(*) - COUNT(`date`) AS missing_date,
       COUNT(*) - COUNT(`children`) AS missing_children,
       COUNT(*) - COUNT(`charges`) AS missing_charges,
       COUNT(*) - COUNT(`Hospital Tier`) AS missing_HospitalTier,
       COUNT(*) - COUNT(`City Tier`) AS missing_CityTier,
       COUNT(*) - COUNT(`State ID`) AS missing_StateID,
       COUNT(*) - COUNT(`BMI`) AS missing_BMI,
       COUNT(*) - COUNT(`HBA1C`) AS missing_HBA1C,
       COUNT(*) - COUNT(`Heart Issues`) AS missing_HeartIssues,
       COUNT(*) - COUNT(`Any Transplants`) AS missing_AnyTransplants,
       COUNT(*) - COUNT(`Cancer History`) AS missing_CancerHistory,
       COUNT(*) - COUNT(`NumberOfMajorSurgeries`) AS
    missing_NumberOfMajorSurgeries,
       COUNT(*) - COUNT(`smoker`) AS missing_smoker,
       COUNT(*) - COUNT(`Name`) AS missing_Name
    FROM MasterData;

-- Handling the trivial details like replacing the missing values with NULL
    UPDATE MasterData
    SET `charges` = NULL
    WHERE `charges` = '?';

    UPDATE MasterData
    SET `BMI` = NULL
    WHERE `BMI` = '?';

    UPDATE MasterData
    SET `HBA1C` = NULL
    WHERE `HBA1C` = '?';

    UPDATE MasterData
    SET `Heart Issues` = NULL
    WHERE `Heart Issues` = '?';

    UPDATE MasterData
    SET `Any Transplants` = NULL
```

```
WHERE `Any Transplants` = '?';

UPDATE MasterData
SET `Cancer History` = NULL
WHERE `Cancer History` = '?';

UPDATE MasterData
SET `NumberOfMajorSurgeries` = NULL
WHERE `NumberOfMajorSurgeries` = '?';

UPDATE MasterData
SET `smoker` = NULL
WHERE `smoker` = '?';

UPDATE MasterData
SET `Name` = NULL
WHERE `Name` = '?';
```

-- Since we are getting a truncate error and we're operating in safe mode we need to disable and then push the same NULL values for all missing rows/columns.

```
SET SQL_SAFE_UPDATES = 0;
UPDATE MasterData
SET `Name` = NULL
WHERE `Name` = '?';

SET SQL_SAFE_UPDATES = 1;

UPDATE MasterData
SET `Name` = NULL
WHERE `Name` = '?'
AND `Customer ID` IS NOT NULL;
```

-- Transform Categorical Variables (Nominal and Ordinal)

```
SELECT
   `Customer ID`,
   CASE
      WHEN `Hospital Tier` = 'Tier 1' THEN 1
      WHEN `Hospital Tier` = 'Tier 2' THEN 2
      WHEN `Hospital Tier` = 'Tier 3' THEN 3
      ELSE NULL
   END AS HospitalTier_Encoded,
   CASE
      WHEN `City Tier` = 'Tier 1' THEN 1
      WHEN `City Tier` = 'Tier 2' THEN 2
```

```
            ELSE NULL
        END AS CityTier_Encoded
    FROM MasterData;

-- Handling Dummy variables in the state.
    SELECT
        `Customer ID`,
        CASE WHEN `State ID` = 'R1011' THEN 1 ELSE 0 END AS State_R1011,
        CASE WHEN `State ID` = 'R1012' THEN 1 ELSE 0 END AS State_R1012,
        CASE WHEN `State ID` = 'R1013' THEN 1 ELSE 0 END AS State_R1013
    FROM MasterData;

-- Clean "NumberOfMajorSurgeries" Column (String Values)
-- If NumberOfMajorSurgeries has string values, we need to convert it into a numeric format.
-- We can remove non-numeric values and then cast the column

    SET SQL_SAFE_UPDATES = 0;

    UPDATE MasterData
    SET `NumberOfMajorSurgeries` = CAST(`NumberOfMajorSurgeries` AS SIGNED)
    WHERE `NumberOfMajorSurgeries` REGEXP '^[0-9]+$';

    SET SQL_SAFE_UPDATES=1;

-- To Handle invalid data
    SET SQL_SAFE_UPDATES = 0;
    UPDATE MasterData
    SET `NumberOfMajorSurgeries` = NULL
    WHERE `NumberOfMajorSurgeries` NOT REGEXP '^[0-9]+$';
    SET SQL_SAFE_UPDATES=1;

-- Calculate Patient Ages Based on Date of Birth
    SELECT
        `Customer ID`,
        TIMESTAMPDIFF(YEAR, `date`, CURDATE()) AS Age
    FROM MasterData;

--  Determine Gender Based on Salutation
    SELECT
        `Customer ID`,
        CASE
            WHEN `Name` LIKE 'Mr%' THEN 'Male'
            WHEN `Name` LIKE 'Ms%' OR `Name` LIKE 'Mrs%' THEN 'Female'
```

```
        ELSE 'Unknown'
    END AS Gender
FROM MasterData;
```

-- Visualize the Distribution of Costs
-- To check the distribution of charges, use SQL aggregation functions
-- While SQL doesn't support complex visualizations, we can calculate the mean, median, and other statistics:

```
SELECT
    MIN(charges) AS Min_Charges,
    MAX(charges) AS Max_Charges,
    AVG(charges) AS Avg_Charges
FROM MasterData;
```

-- median

-- First query for Min, Max, Avg

```
SELECT
    MIN(charges) AS Min_Charges,
    MAX(charges) AS Max_Charges,
    AVG(charges) AS Avg_Charges
FROM MasterData;
```

-- Second query for Median calculation

```
SELECT COUNT(*) AS TotalCount FROM MasterData;
```

-- Calculate Median (For Odd and Even Counts)

```
SET @Offset = FLOOR(@TotalCount / 2);

SET @Query = CONCAT(
    'SELECT charges AS Median_Charges FROM MasterData ORDER BY charges
LIMIT 1 OFFSET ',
    @Offset
);
PREPARE stmt FROM @Query;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;

SELECT @@sql_mode;
SET sql_mode = '';
SELECT @Query;
```

```
SELECT VERSION();

SET @Offset = (SELECT FLOOR(COUNT(*) / 2) FROM MasterData);

SET @TotalCount = (SELECT COUNT(*) FROM MasterData);
SET @Offset = FLOOR(@TotalCount / 2);

SET @Query = CONCAT(
   'SELECT charges AS Median_Charges FROM MasterData ORDER BY charges
LIMIT 1 OFFSET ',
   @Offset
);

-- Prepare the dynamic query
PREPARE stmt FROM @Query;

-- Execute the query
EXECUTE stmt;

-- Deallocate the prepared statement
DEALLOCATE PREPARE stmt;

-- visualize the charts
-- Histogram
SELECT FLOOR(Charges/1000)*1000 AS ChargeRange, COUNT(*) AS Frequency
FROM MasterData
GROUP BY FLOOR(Charges/1000)*1000;

-- Extract Gender from Salutaitons
ALTER TABLE MasterData ADD COLUMN Gender VARCHAR(10);
SET SQL_SAFE_UPDATES = 0;
UPDATE MasterData
SET Gender = CASE
   WHEN Name LIKE 'Mr.%' THEN 'Male'
   WHEN Name LIKE 'Mrs.%' OR Name LIKE 'Ms.%' THEN 'Female'
   ELSE 'Unknown' END;
SET SQL_SAFE_UPDATES = 1;

Select Gender from MasterData;

-- Gender based boxplot
SELECT Gender, MIN(Charges) AS MinCharge, MAX(Charges) AS MaxCharge,
AVG(Charges) AS AvgCharge
FROM MasterData
```

```
    GROUP BY Gender;

-- Cost distribution by gender and hospital tier
    SELECT Gender, `Hospital tier`, AVG(Charges) AS AvgCharge
    FROM MasterData
    GROUP BY Gender, `Hospital tier`;

-- Radar chart for median costs by hospital tier
    SELECT
      `Customer ID`,
      CASE
        WHEN `Hospital tier` = 'Tier 1' THEN 1
        WHEN `Hospital tier` = 'Tier 2' THEN 2
        WHEN `Hospital tier` = 'Tier 3' THEN 3
        ELSE NULL
      END AS HospitalTier_Encoded,
      CASE
        WHEN `City tier` = 'Tier 1' THEN 1
        WHEN `City tier` = 'Tier 2' THEN 2
        ELSE NULL
      END AS CityTier_Encoded
    FROM MasterData;

    DESCRIBE MasterData;

    SET SQL_SAFE_UPDATES = 0;

    UPDATE MasterData
    SET HospitalTier_Encoded =
      CASE
        WHEN `Hospital Tier` = 'Tier 1' THEN 1
        WHEN `Hospital Tier` = 'Tier 2' THEN 2
        WHEN `Hospital Tier` = 'Tier 3' THEN 3
        ELSE NULL
      END;

    UPDATE MasterData
    SET HospitalTier_Encoded =
      CASE
        WHEN HospitalTier = 'Tier 1' THEN 1
        WHEN HospitalTier = 'Tier 2' THEN 2
        WHEN HospitalTier = 'Tier 3' THEN 3
        ELSE NULL
      END;
```

```
SET SQL_SAFE_UPDATES = 1;

SELECT
   CityTier_Encoded,
   HospitalTier_Encoded,
   COUNT(*) AS Count
FROM MasterData
GROUP BY CityTier_Encoded, HospitalTier_Encoded;

DESCRIBE MasterData;
ALTER TABLE MasterData ADD CityTier_Encoded INT;
UPDATE MasterData
SET CityTier_Encoded =
   CASE
      WHEN `City Tier` = 'Tier 1' THEN 1
      WHEN `City Tier` = 'Tier 2' THEN 2
      WHEN `City Tier` = 'Tier 3' THEN 3
      ELSE NULL
   END;

SELECT
   CityTier_Encoded,
   HospitalTier_Encoded,
   COUNT(*) AS Count
FROM MasterData
GROUP BY CityTier_Encoded, HospitalTier_Encoded
LIMIT 0, 10000;

SELECT `City Tier`, `Hospital Tier`, COUNT(*) AS Count
FROM MasterData
GROUP BY `City Tier`, `Hospital Tier`;

-- ANOVA testing
SELECT `Hospital Tier`, Charges
FROM MasterData;

-- For t-Tests
SELECT Smoker, Charges
FROM MasterData;

-- For Chi-Sqaured Independednce
SELECT Smoker, `Heart Issues`, COUNT(*) AS Count
FROM MasterData
```

```
GROUP BY Smoker, `Heart Issues`;

SHOW VARIABLES LIKE 'secure_file_priv';
```

-- export the files to computer

```
SELECT *
INTO OUTFILE '/Users/shivanimishra/Documents/IIT - K /Processed Captsone
Data/masterdata.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY ""
LINES TERMINATED BY '\n'
FROM MasterData;

Show grants for current_user;
GRANT FILE ON *.* TO 'your_username'@'localhost';
FLUSH PRIVILEGES;
```

-- WEEK 02 MACHINE LEARNING PART

-- SQL Task
-- 1. Merge Tables

-- Steps:
-- Identify a common column for merging, such as Patient_ID.
-- Remove duplicates and null values:

```
SELECT * from hospitalisationdetails;
select * from medicalexaminations;
```

-- Assuming the two tables are 'hospitalisation_details' and 'medical_examinations'

```
SELECT *
FROM hospitalisationdetails h
JOIN medicalexaminations m
   ON h.`Customer ID` = m.`Customer ID`;
```

-- 2. Add Primary Key Constraints
-- You need to ensure that Customer_ID is unique in both tables. This can be done by removing any duplicates or NULL values in the Customer_ID column and then adding a primary key constraint.

-- SQL Code for Removing Duplicates and Adding Primary Key:
-- First, remove duplicates and NULL values in the Customer_ID column
-- Remove duplicates from hospitalization_details table

```
-- Step 1: Create a temporary table to store the Customer IDs with duplicates
    CREATE TEMPORARY TABLE temp_ids AS
    SELECT `Customer ID`
    FROM medicalexaminations
    GROUP BY `Customer ID`
    HAVING COUNT(*) > 1;

-- deactivate safe mode first
    SET SQL_SAFE_UPDATES = 0;

-- Step 2: Delete from medicalexaminations using the temporary table
    DELETE FROM medicalexaminations
    WHERE `Customer ID` IN (SELECT `Customer ID` FROM temp_ids);

-- Step 3: Drop the temporary table
    DROP TEMPORARY TABLE temp_ids;

-- re-activate safe mode
    SET SQL_SAFE_UPDATES = 1;

-- Add primary key constraints
    ALTER TABLE hospitalisationdetails
    ADD PRIMARY KEY (`Customer ID`);

    ALTER TABLE hospitalisationdetails
    MODIFY COLUMN `Customer ID` VARCHAR(255);  -- Or use INT if appropriate

    describe hospitalisationdetails;

    ALTER TABLE medicalexaminations
    ADD PRIMARY KEY (`Customer ID`);

    ALTER TABLE medicalexaminations
    MODIFY COLUMN `Customer ID` VARCHAR(255);  -- Or use INT if appropriate

    DESCRIBE hospitalisationdetails;
    describe medicalexaminations;

-- checking for null values
    UPDATE hospitalisationdetails
    SET `Customer ID` = 'default_value'
    WHERE `Customer ID` IS NULL;

    SET SQL_SAFE_UPDATES = 0;
```

```
SET SQL_SAFE_UPDATES = 1;

UPDATE hospitalisationdetails
SET `Customer ID` = 'default_value'
WHERE `Customer ID` IS NULL
AND `Customer ID` IS NOT NULL;

SHOW INDEX FROM hospitalisationdetails;

SELECT *
FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS
WHERE TABLE_NAME = 'hospitalisationdetails'
  AND CONSTRAINT_TYPE = 'PRIMARY KEY';

SHOW INDEX FROM medicalexaminations;
```

-- Retrieve Information about People who are Diabetic and Have Heart Problems
-- To retrieve the data for people who are diabetic (HBA1C > 6.5) and have heart problems, along with their average age, number of children, BMI, and hospitalization costs:

-- Given that you have separate columns for year, month, and day in your hospitalisationdetails table
-- We can create a date by combining these columns and then calculate the age.
-- Assuming the columns are named year, month, and date, you can use the following SQL query to calculate the average age:

```
SELECT
   AVG(DATEDIFF(CURRENT_DATE, STR_TO_DATE(CONCAT(h.year, '-', h.month, '-',
h.date), '%Y-%m-%d')) / 365) AS avg_age,
   AVG(h.children) AS avg_children,
   AVG(m.BMI) AS avg_bmi,
   AVG(h.charges) AS avg_hospitalization_cost
FROM
   hospitalisationdetails h
JOIN
   medicalexaminations m ON h.`Customer ID` = m.`Customer ID`
WHERE
   m.HBA1C > 6.5 AND m.`Heart Issues` = 'Yes';
```

-- describe hospitalisationdetails

-- 4. Find the Average Hospitalization Cost for Each Hospital Tier and City Level
-- To determine the average hospitalization cost for each hospital tier and city level:

```
SELECT
    h.`Hospital tier`,
    h.`City tier`,
    AVG(h.charges) AS avg_hospitalization_cost
FROM
    hospitalisationdetails h
JOIN
    medicalexaminations m
    ON h.`Customer ID` = m.`Customer ID`
GROUP BY
    h.`Hospital tier`, h.`City tier`;
```

-- 5. Determine the Number of People Who Have Had Major Surgery with a History of Cancer
-- To determine the number of people who have had major surgery and have a history of cancer, use the following query:

```
SELECT
    COUNT(*) AS num_people_with_surgery_and_cancer
FROM
    medicalexaminations m
JOIN
    hospitalisationdetails h
    ON h.`Customer ID` = m.`Customer ID`
WHERE
    m.NumberOfMajorSurgeries > 0
    AND m.`Cancer history` = 'Yes';
```

-- 6. Determine the Number of Tier-1 Hospitals in Each State
-- To count the number of tier-1 hospitals in each state:

```
SELECT
    h.`State ID`,
    COUNT(*) AS num_tier_1_hospitals
FROM
    hospitalisationdetails h
WHERE
    h.`Hospital tier` = 'tier-1'
GROUP BY
    h.`State ID`;
```

```
SELECT DISTINCT `Hospital tier` FROM hospitalisationdetails;
SELECT `Hospital tier`, `State ID` FROM hospitalisationdetails WHERE `Hospital tier` =
'tier-1' LIMIT 10;
SELECT COUNT(*) FROM hospitalisationdetails WHERE `State ID` IS NULL;
SELECT * FROM hospitalisationdetails WHERE `Hospital tier` = 'tier-1' LIMIT 10;

SELECT
   h.`State ID`,
   COUNT(*) AS num_tier_1_hospitals
FROM
   hospitalisationdetails h
WHERE
   h.`Hospital tier` = 'tier-1'  -- Ensure 'tier-1' exists exactly as it is in the database
GROUP BY
   h.`State ID`;
```