SQL QUERIES ON CREDIT CARD FRAUD PREDICTION:-

Data Exploration

To explore the data using SQL, you can use the following query to retrieve the first 10 rows of the dataset:

Query:-

SELECT * FROM creditcard LIMIT 10;

To get the summary statistics for each column, you can use the following query:

Query:-

```
SELECT COLUMN_NAME,
AVG(VALUE) AS mean,
STDDEV_SAMP(VALUE) AS std,
MIN(VALUE) AS min,
0.25 * PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY VALUE) AS q25, 0.5 *
PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY VALUE) AS q50, 0.75 *
PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY VALUE) AS q75, MAX(VALUE)
AS max FROM creditcard GROUP BY COLUMN_NAME;
```

Data Preprocessing

To load the data into a SQL database, you would need to create a table with the same structure as the CSV file. Assuming the CSV file is named `creditcard.csv` and is located in the `/path/to/data/` directory, you can create the table using the following SQL query:

```
sql
CREATE TABLE creditcard (
  Time FLOAT,
  V1 FLOAT,
  V2 FLOAT,
  V3 FLOAT,
  V4 FLOAT,
  V5 FLOAT,
  V6 FLOAT,
  V7 FLOAT,
  V8 FLOAT,
  V9 FLOAT,
  V10 FLOAT,
  V11 FLOAT,
  V12 FLOAT,
  V13 FLOAT,
  V14 FLOAT,
  V15 FLOAT,
  V16 FLOAT,
  V17 FLOAT,
  V18 FLOAT,
```

```
  V19 FLOAT,
  V20 FLOAT,
  V21 FLOAT,
  V22 FLOAT,
  V23 FLOAT,
  V24 FLOAT,
  V25 FLOAT,
  V26 FLOAT,
  V27 FLOAT,
  V28 FLOAT,
  Class INT
);

COPY creditcard FROM '/path/to/data/creditcard.csv' DELIMITER ',' CSV HEADER;
```

To standardize the features, you can create a view that applies the standardization:

```
sql
CREATE VIEW standardized_features AS
SELECT
  Time,
  (V1 - AVG(V1)) / STDDEV_SAMP(V1) AS std_V1,
  (V2 - AVG(V2)) / STDDEV_SAMP(V2) AS std_V2,
  (V3 - AVG(V3)) / STDDEV_SAMP(V3) AS std_V3,
  (V4 - AVG(V4)) / STDDEV_SAMP(V4) AS std_V4,
  (V5 - AVG(V5)) / STDDEV_SAMP(V5) AS std_V5,
  (V6 - AVG(V6)) / STDDEV_SAMP(V6) AS std_V6,
  (V7 - AVG(V7)) / STDDEV_SAMP(V7) AS std_V7,
  (V8 - AVG(V8)) / STDDEV_SAMP(V8) AS std_V8,
  (V9 - AVG(V9)) / STDDEV_SAMP(V9) AS std_V9,
  (V10 - AVG(V10)) / STDDEV_SAMP(V10) AS std_V10,
  (V11 - AVG(V11)) / STDDEV_SAMP(V11) AS std_V11,
  (V12 - AVG(V12)) / STDDEV_SAMP(V12) AS std_V12,
  (V13 - AVG(V13)) / STDDEV_SAMP(V13) AS std_V13,
  (V14 - AVG(V14)) / STDDEV_SAMP(V14) AS std_V14,
  (V15 - AVG(V15)) / STDDEV_SAMP(V15) AS std_V15,
  (V16 - AVG(V16)) / STDDEV_SAMP(V16) AS std_V16,
  (V17 - AVG(V17)) / STDDEV_SAMP(V17) AS std_V17,
  (V18 - AVG(V18)) / STDDEV_SAMP(V18) AS std_V18,
  (V19 - AVG(V19)) / STDDEV_SAMP(V19) AS std_V19,
  (V20 - AVG(V20)) / STDDEV_SAMP(V20) AS std_V20,
  (V21 - AVG(V21)) / STDDEV_SAMP(V21) AS std_V21,
  (V22 - AVG(V22)) / STDDEV_SAMP(V22) AS std_V22,
  (V23 - AVG(V23)) / STDDEV_SAMP(V23) AS std_V23,
  (V24 - AVG(V24)) / STDDEV_SAMP(V24) AS std_V24,
  (V25 - AVG(V25)) / STDDEV_SAMP(V25) AS std_V25,
  (V26 - AVG(V26)) / STDDEV_SAMP(V26) AS std_V26,
  (V27 - AVG(V27))
```

Modeling

To build a logistic regression model, you can use the following SQL query:

```
sql
CREATE TABLE logistic_regression_features AS
SELECT
  std_V1,
  std_V2,
  std_V3,
  std_V4,
  std_V5,
  std_
```

CREATE VIEW standardized_features AS SELECT std_V1, std_V2, std_V3, std_V4, std_V5, std_V6, std_V7, std_V8, std_V9, std_V20, -- Assuming V20 is the target column Class FROM (SELECT Time, std(V1) AS std_V1, std(V2) AS std_V2, std(V3) AS std_V3, std(V4) AS std_V4, std(V5) AS std_V5, std(V6) AS std_V6, std(V7) AS std_V7, std(V8) AS std_V8, std(V9) AS std_V9, (V20 - AVG(V20)) / STDDEV_SAMP(V20) AS std_V20, Class FROM creditcard GROUP BY Time) t;

FOR LOGISTIC REGRESSION VIEW:-

CREATE TABLE logistic_regression_model AS SELECT std_V1, std_V2, std_V3, std_V4, std_V5, std_V6, std_V7, std_V8, std_V9, std_V20 FROM standardized_features; CREATE TABLE logistic_regression_model_predictions AS SELECT std_V1, std_V2, std_V3, std_V4, std_V5, std_V6, std_V7, std_V8, std_V9, std_V20, CASE WHEN logistic_regression_model.prediction > 0.5 THEN 1 ELSE 0 END AS prediction FROM (SELECT std_V1, std_V2, std_V3, std_V4, std_V5, std_V6, std_V7, std_V8, std_V9, std_V20, PROBABILITY(Class = 1) OVER (PARTITION BY std_V1, std_V2, std_V3, std_V4, std_V5, std_V6, std_V7, std_V8, std_V9, std_V20) AS prediction FROM logistic_regression_model) t;

To evaluate the model, you can use the following SQL:

```
sql
SELECT
  precision_score(Class, prediction) AS precision,
  recall_score(Class, prediction) AS recall,
  f1_score(Class, prediction) AS f1,
  roc_auc_score(Class, prediction) AS roc_auc
FROM logistic_regression_model_predictions;
```