**SQL Documentation for Insurance Claims Analysis**

**1. Data Cleaning Steps**

**1.1 Validate Policy Numbers**

**Objective**: Ensure all policy\_number values are exactly six digits.

SELECT policy\_number

FROM

 `custom-zone-444222-i8.Insurance\_claims.insurance\_claims`

 WHERE LENGTH(CAST(policy\_number AS STRING)) != 6

**1.2 Identify Missing Policy Numbers**

**Objective**: Detect records with NULL values in the policy\_number field.

SELECT policy\_number

FROM

 `custom-zone-444222-i8.Insurance\_claims.insurance\_claims`

 WHERE policy\_number is null

* 1. **Clean Negative Umbrella Limits**

**Objective**: Exclude negative values in the umbrella\_limit field and create a cleaned column.

SELECT umbrella\_limit AS new\_umbrella\_limit

FROM

 `custom-zone-444222-i8.Insurance\_claims.insurance\_claims`

 WHERE umbrella\_limit >= 0

**2. Data Formatting**

**2.1 Split policy\_csl into Liability and Umbrella Coverage**

**Objective**: Parse the policy\_csl field to separate liability and umbrella coverage values.

SELECT

SPLIT(policy\_csl,'/')[OFFSET(0)] AS liability\_coverage,

SPLIT(policy\_csl,'/')[OFFSET(1)] AS umbrella\_coverage

FROM

 `custom-zone-444222-i8.Insurance\_claims.insurance\_claims`

 WHERE umbrella\_limit >= 0

**3. Descriptive Statistics and Exploratory Data Analysis (EDA)**

**3.1 Compute Policy Premium Statistics**

**Objective**: Calculate the average, median, and standard deviation for policy\_annual\_premium.

SELECT

AVG(policy\_annual\_premium) AS avg\_policy\_annual\_premium,

APPROX\_QUANTILES(policy\_annual\_premium,2)[OFFSET(1)] AS median\_poilcy\_annual\_premium,

STDDEV(policy\_annual\_premium) AS stddev\_policy\_annual\_premium

FROM

`custom-zone-444222-i8.Insurance\_claims.insurance\_claims`

**3.2 Analyze Correlation Between Fraud and Policy Types**

**Objective**: Evaluate the fraud rate for each type of policy\_csl.

SELECT

  policy\_csl,

  AVG

  (CASE WHEN fraud\_reported = TRUE THEN 1

  ELSE 0

  END) AS fraud\_rate

FROM

  `custom-zone-444222-i8.Insurance\_claims.insurance\_claims`

GROUP BY policy\_csl

**4. Risk-Based Data Profiling**

**4.1 Create Risk Segments**

**Objective**: Categorize risk based on policy\_annual\_premium and age

SELECT

  policy\_number,

  age,

  policy\_annual\_premium,

  CASE

   WHEN policy\_annual\_premium > 1500 AND age BETWEEN 55 AND 64 THEN 'High Risk'

WHEN policy\_annual\_premium BETWEEN 1000 AND 1500 AND age BETWEEN 35 AND 54 THEN 'Medium Risk'

WHEN policy\_annual\_premium <= 1000 AND age < 25 THEN 'High Risk' -- Adding condition for age under 25

WHEN policy\_annual\_premium BETWEEN 1000 AND 1500 AND age BETWEEN 25 AND 34 THEN 'Medium Risk'

WHEN policy\_annual\_premium <= 1000 AND age >= 25 AND age <= 34 THEN 'Low Risk'

    ELSE 'N/A'

  END AS risk\_category

FROM

  `custom-zone-444222-i8.Insurance\_claims.insurance\_claims`

**4.2 Identify Fraud Red Flags**

**Objective**: Count non-fraudulent claims (fraud\_reported = FALSE) by incident\_type.

SELECT incident\_type,

SUM(

  CASE

  WHEN fraud\_reported = FALSE THEN 1

  ELSE 0

  END

) AS fraud\_count

FROM

`custom-zone-444222-i8.Insurance\_claims.insurance\_claims`

GROUP By incident\_type

**5. Visualizations**

**5.1 Incident Type vs Fraud Ratio**

**Objective**: Compare the fraud ratio for each incident\_type.

SELECT incident\_type,

SUM( CASE WHEN fraud\_reported = TRUE THEN 1 ELSE 0 END) AS fraud\_count\_y,

SUM( CASE WHEN fraud\_reported = FALSE THEN 1 ELSE 0 END) AS fraud\_count\_n,

SUM( CASE WHEN fraud\_reported = TRUE THEN 1 ELSE 0 END)/ NULLIF(SUM( CASE WHEN fraud\_reported = FALSE THEN 1 ELSE 0 END),0) AS fraud\_ratio

FROM

`custom-zone-444222-i8.Insurance\_claims.insurance\_claims`

GROUP By incident\_type

#### A graph with blue squares Description automatically generated

#### **5.2 Age Category vs Fraud Counts**

**Objective**: Analyze fraud counts for each age category.

SELECT

CASE

  WHEN age < 25 THEN 'young\_drivers\_25'

  WHEN age BETWEEN 25 AND 34 THEN 'drivers\_25\_34'

  WHEN age BETWEEN 35 AND 44 THEN 'drivers\_35\_44'

  WHEN age BETWEEN 45 AND 54 THEN 'drivers\_45\_54'

  WHEN age BETWEEN 55 AND 64 THEN 'drivers\_55\_64'

  ELSE 'senior\_drivers\_65'

END AS age\_category,

SUM( CASE WHEN fraud\_reported = TRUE THEN 1 ELSE 0 END) AS fraud\_count\_y,

SUM( CASE WHEN fraud\_reported = FALSE THEN 1 ELSE 0 END) AS fraud\_count\_n,

SUM( CASE WHEN fraud\_reported = TRUE THEN 1 ELSE 0 END)/ NULLIF(SUM( CASE WHEN fraud\_reported = FALSE THEN 1 ELSE 0 END),0) AS fraud\_ratio

FROM

`custom-zone-444222-i8.Insurance\_claims.insurance\_claims`

GROUP By age\_category

A graph with blue and green squares

Description automatically generated

**5.3 Risk Analysis over Age:**

**Objective:** Analyze risk over different Age groups.

SELECT

  age,

  CASE

   WHEN policy\_annual\_premium > 1500 AND age BETWEEN 55 AND 64 THEN 'High Risk'

    WHEN policy\_annual\_premium BETWEEN 1000 AND 1500 AND age BETWEEN 35 AND 54 THEN 'Medium Risk'

    WHEN policy\_annual\_premium <= 1000 AND age < 25 THEN 'High Risk' -- Adding condition for age under 25

    WHEN policy\_annual\_premium BETWEEN 1000 AND 1500 AND age BETWEEN 25 AND 34 THEN 'Medium Risk'

    WHEN policy\_annual\_premium <= 1000 AND age >= 25 AND age <= 34 THEN 'Low Risk'

    ELSE 'No Risk'

  END AS risk\_category

FROM

  `custom-zone-444222-i8.Insurance\_claims.insurance\_claims`

GROUP BY policy\_number,age,policy\_annual\_premium

A graph with blue squares

Description automatically generated

**5.4 Total Claims by Incident Type:**

**Objective**: Examine claims for each type of incident

SELECT

  incident\_type,

  COUNT(policy\_number) AS total\_claims

FROM `custom-zone-444222-i8.Insurance\_claims.insurance\_claims`

GROUP BY incident\_type

ORDER BY total\_claims DESC;

A graph with blue squares

Description automatically generated

|  |  |
| --- | --- |
| **incident\_type** | **total\_claims** |
| Multi-vehicle Collision | 419 |
| Single Vehicle Collision | 403 |
| Vehicle Theft | 94 |
| Parked Car | 84 |

**5.5 Time Trend of Fraudulent Claims**

**Objective**: Examine fraud trends over time.

SELECT

  DATE\_TRUNC(incident\_date, MONTH) AS claim\_month,

  SUM(CASE WHEN fraud\_reported = TRUE THEN 1 ELSE 0 END) AS fraud\_count\_y,

  SUM(CASE WHEN fraud\_reported = FALSE THEN 1 ELSE 0 END) AS fraud\_count\_n

FROM `custom-zone-444222-i8.Insurance\_claims.insurance\_claims`

GROUP BY claim\_month

ORDER BY claim\_month;

|  |  |  |
| --- | --- | --- |
| **claim\_month** | **fraud\_count\_y** | **fraud\_count\_n** |
| 1/1/2015 | 135 | 381 |
| 2/1/2015 | 108 | 364 |
| 3/1/2015 | 4 | 8 |

#### **5.6 Fraud Percentage by Age Group**

**Objective**: Calculate the fraud percentage for each age group.

SELECT

  CASE

    WHEN age < 25 THEN 'Young Drivers (<25)'

    WHEN age BETWEEN 25 AND 34 THEN 'Drivers 25-34'

    WHEN age BETWEEN 35 AND 44 THEN 'Drivers 35-44'

    WHEN age BETWEEN 45 AND 54 THEN 'Drivers 45-54'

    WHEN age BETWEEN 55 AND 64 THEN 'Drivers 55-64'

    ELSE 'Senior Drivers (65+)'

  END AS age\_category,

  COUNT(policy\_number) AS total\_claims,

  SUM(CASE WHEN fraud\_reported = TRUE THEN 1 ELSE 0 END) AS fraud\_count\_y,

  SAFE\_DIVIDE(

    SUM(CASE WHEN fraud\_reported = TRUE THEN 1 ELSE 0 END),

    COUNT(policy\_number)

  ) \* 100 AS fraud\_percentage

FROM `custom-zone-444222-i8.Insurance\_claims.insurance\_claims`

GROUP BY age\_category

ORDER BY fraud\_percentage DESC;

|  |  |  |  |
| --- | --- | --- | --- |
| **age\_category** | **total\_claims** | **fraud\_count\_y** | **fraud\_percentage** |
| Drivers 55-64 | 78 | 25 | 32.05 |
| Young Drivers (<25) | 26 | 8 | 30.76 |
| Drivers 25-34 | 334 | 88 | 26.34 |
| Drivers 35-44 | 391 | 88 | 22.50 |
| Drivers 45-54 | 171 | 38 | 22.22 |

**5.7 Claims Severity Analysis**

**Objective:** Analyze premiums for fraudulent vs. non-fraudulent claims.

SELECT

  fraud\_reported,

  MIN(policy\_annual\_premium) AS min\_premium,

  MAX(policy\_annual\_premium) AS max\_premium,

  AVG(policy\_annual\_premium) AS avg\_premium,

  STDDEV(policy\_annual\_premium) AS stddev\_premium

FROM `custom-zone-444222-i8.Insurance\_claims.insurance\_claims`

GROUP BY fraud\_reported

ORDER BY fraud\_reported;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **fraud\_reported** | **min\_premium** | **max\_premium** | **avg\_premium** | **stddev\_premium** |
| FALSE | 433.33 | 2047.59 | 1258.43 | 241.2472104 |
| TRUE | 484.67 | 1935.85 | 1250.236275 | 253.2636509 |