CREDIT CARD FRAUDULENT TRANSACTION ANALYSIS: CASE STUDY USING SQL

It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

For this case study, we have taken the dataset from kaggle. The dataset consists of all information on the Card Details,Customer Details ,Transaction and Fraudulent transaction details for the year 2016

Kaggle dataset link

https://www.kaggle.com/datasets/ananta/credit-card-data?

Using the Credit Card data dataset, solved the following problems

1) How many customers have done transactions over 49000?

```
select count(distinct cust_id) as no_of_customers
from Transaction_base trn
join Card_base crd on trn.credit_card_id = crd.card_number
where trn.transaction_value > 49000;
```

2) What kind of customers can get a Premium credit card?

```
select  distinct customer_segment
from Card_base crd
join Customer_base cst on cst.cust_id = crd.cust_id
where card_family = 'Premium';
```

3) Identify the range of credit limit of customer who have done fraudulent transactions?

```
select min(credit_limit), max(credit_limit)
from Transaction_base trn
join Fraud_base frd on frd.transaction_id=trn.transaction_id
join Card_base crd on trn.credit_card_id = crd.card_number;
```

4) What is the average age of customers who are involved in fraud transactions based on different card type?

```
select card_family, avg(age)
from Transaction_base trn
join Fraud_base frd on frd.transaction_id=trn.transaction_id
join Card_base crd on trn.credit_card_id = crd.card_number
join customer_base cst on cst.cust_id=crd.cust_id
group by card_family
```

5) Identify the month when highest no of fraudulent transactions occured.

```
select to_char(transaction_date,'MON') as mon, count(1) as no_of_fraud_trns
from Transaction_base trn
join Fraud_base frd on frd.transaction_id=trn.transaction_id
group by to_char(transaction_date,'MON')
order by no_of_fraud_trns desc
limit 1;
```

6) Identify the customer who has done the most transaction value without involving in any fraudulent transactions.

```
select cst.cust_id, sum(trn.transaction_value) as total_trns
from Transaction_base trn
join Card_base crd on trn.credit_card_id = crd.card_number
join customer_base cst on cst.cust_id=crd.cust_id
where cst.cust_id not in
    (select crd.cust_id
     from Transaction_base trn
    join Fraud_base frd on frd.transaction_id=trn.transaction_id
    join Card_base crd on trn.credit_card_id = crd.card_number)
group by cst.cust_id
order by total_trns desc
limit 1;
```

7) Check and return any customers who have not done a single transaction.

```
select distinct cust_id
from customer_base cst
where cst.cust_id not in
  (select distinct crd.cust_id
   from Transaction_base trn
  join Card_base crd on trn.credit_card_id = crd.card_number);
```

8) What is the highest and lowest credit limit given to each card type?

```
select card_family, max(credit_limit) max_limit, min(credit_limit) min_limit
from Card_base
group by card_family;
```

9) What is the total value of transactions done by customers who come under the age bracket of 20-30 yrs, 30-40 yrs, 40-50 yrs, 50+ yrs and 0-20 yrs.

```
select sum(case when age > 0 and age <= 20 then transaction_value else 0 end) as
trns_value_0_to_20
, sum(case when age > 20 and age <= 30 then transaction_value else 0 end) as
trns_value_20_to_30
, sum(case when age > 30 and age <= 40 then transaction_value else 0 end) as
trns_value_30_to_40
, sum(case when age > 40 and age <= 50 then transaction_value else 0 end) as
trns_value_40_to_50
```

```
        , sum(case when age > 50 then transaction_value else 0 end) as
trns_value_greater_than_50
        from Transaction_base trn
        join Card_base crd on trn.credit_card_id = crd.card_number
        join customer_base cst on cst.cust_id=crd.cust_id
```

10) Which card type has done the most no of transactions and the total highest value of transactions without having any fraudulent transactions.

```
        select *
        from (
                select card_family, count(1) as val, 'Highest number of trns' desc
                from Transaction_base trn
                join Card_base crd on trn.credit_card_id = crd.card_number
                where crd.card_family not in
                   (select distinct crd.card_family
                    from Transaction_base trn
                    join Fraud_base frd on frd.transaction_id=trn.transaction_id
                   join Card_base crd on trn.credit_card_id = crd.card_number)
                group by card_family
                order by val desc
                limit 1) x
        union all
        select * from (
                select card_family, sum(transaction_value) as val, 'Highest value of trns' desc
                from Transaction_base trn
                join Card_base crd on trn.credit_card_id = crd.card_number
                where crd.card_family not in
                (select distinct crd.card_family
                from Transaction_base trn
                join Fraud_base frd on frd.transaction_id=trn.transaction_id
                join Card_base crd on trn.credit_card_id = crd.card_number)
                group by card_family
                order by val desc
                limit 1) y
```