SET - 3

Q1: You have two mysql tables:

players (player_id integer, name varchar(255), country varchar(255))
match_scores (match_id integer, player_id integer, runs integer, balls integer, fours integer, sixes integer)

You can assume match_id in the order the matches have been played. So match_id 1 has been played before the match_id 2, and so on.All match IDs are integers. Match ID 5 was played between Australia and South Africa.

Write sql queries to find
(a) what is the batting average for "Rohit Sharma" where Batting Average is defined as "Overall runs scored in the career" divided by "Total number of matches in which he has batted".

```
SELECT  SUM(runs)/COUNT(match_id) as Batting_avg
FROM match_scores
INNER JOIN players
ON match_scores.player_id = players.player_id
WHERE players.name = "Rohit Sharma"
```

Assuming runs as total runs scored in a match

(b) how many matches did "Rohit Sharma" play before scoring his first double century.

```
With rohit_2centuries (match_id, runs, match_played)
as(
SELECT match_scores.match_id, match_scores.runs, row_number() OVER (ORDER BY match_id) as
match_played
FROM match_scores
INNER JOIN players
ON match_scores.player_id = players.player_id
WHERE players.name = "Rohit Sharma"
AND match_scores.runs >= 200
)
SELECT MIN(match_played) - 1
FROM rohit_2centuries
```

Assuming runs as total runs scored in a match

Q2: Given the tables sms_data, user_loan and payments (with 1 sample data each)

sms_data	**sample**		**user_loan**	**sample**		**payments**	**sample**
ID	27,588,463		**ID**	1440,274		**ID**	7120,001
USER_ID	2,257,841		**USER_ID**	2,257,841		**USER_ID**	2,257,841
SMS_DATE	1,499,680,122,693		**CREATED_AT**	21/04/18 17:03		**CREATED_AT**	23/04/18 03:43
SENDER	obcbnk		**UPDATED_AT**	23/04/18 05:31		**UPDATED_AT**	23/04/18 05:31
TXN_TYPE	DEBIT		**APPLIED_AT**	21/04/18 17:03		**AMOUNT**	3,090
AMOUNT	1,028.00		**AMOUNT**	3,000		**USER_LOAN_ID**	1440,274
ACCOUNT_NUMBER	xxxxx2834		**PURPOSE**	bill payment		**STATUS**	DONE
BALANCE	0		**LOAN_TYPE**	PERSONAL		**MODE**	NEFT
			STATUS	COMPLETED		**REF_NUMBER**	811003100635
			EMI_DATE	23		**PAYMENT_DEADLINE**	23/05/18 03:43
			CLIENT_ID	3		**PAYMENT_DATE**	23/04/18 05:01
			DISBURSED_AT	23/04/18 02:43		**CLIENT_ID**	
			LENDER_ID	10		**TYPE**	REPAYMENT

Note: **'ID'** is the Primary Key for all the above tables.
For example - ID in user_loan table is the user_loan_id

sms_data: contains parsed data of user's sms from his/her mobile device.

user_loan: contains all the loans applied by the user. "Status" shows if it was **'REJECTED'** - loan was rejected, **'ACTIVE'** - loan amount has been transferred to user account, and **'COMPLETED'** - user has paid back and closed the loan).
Note:- There can be multiple loan records for a user as he can avail multiple loans. But there will not be any overlap of time-period in these loans. A user gets a loan only after completion of the previous loan.

payments: contains all the emi payments associated with user_loan. So for every loan there will be multiple EMI payment entries in the payments table
Note**:-** For a loan of 6 months with monthly emi - there will be 6 different records in payments table related to that loan with amount as the emi amount to be paid.
"status" can be 'PENDING' (user has not completed the EMI payment) or 'DONE' (user has paid on the payment_date - it may be delayed).

Payment delay: A payment is considered to be delayed if the payment is not completed before payment deadline. If any user completes the EMI payment before the payment deadline the delayed days is 0.
Principal Amount: The amount w.r.t the loan_id in the user_loan table
EMI Amount: The amount w.r.t payment_id in the payments table.
note:- EMI Amount = (Principal + Interest)/(no of EMIs)
n_day_default_amount: Total principal part of the EMI amount which has not been completed before the n days past the payment deadline. ***= user_loan.amount – no of emi's * emi's***
%Gross_n_days_default: (n_day_default_amount/loan_amount)*100

Write SQL queries to find:

1.avg debit amount in the sms data over the last 30 days from today for user_id=1

```
SELECT SUM(amount)/30
FROM sms_data
WHERE user_id = 1
AND txn_type = 'DEBIT'
AND sms_date between unix_timestamp(date_add(curdate(), interval - 30 day))*1000 and
unix_timestamp(curdate()*1000)
```

Assuming sms_date is in milliseconds from epoch

2.avg payment delayed days over all the completed loans for user_id=1

```
SELECT SUM(CASE WHEN DATEDIFF(P.payment_deadline,P.payment_date) > 0 THEN 0 ELSE
DATEDIFF(P.payment_date, P.payment_deadline)END )/COUNT(DISTINCT L.id )
FROM payments  P
LEFT JOIN user_loan L
ON P.user_loan_id = L.id
WHERE P.user_id = 1
AND L.status = 'COMPLETED'
```

3.emi count which has been repaid for the active loan for the user_id=1 (return -1 if there is no active loan).

```
set @active_loan_count = 0;
select @active_loan_count = count(*) from user_loan where user_id = 1 and status = 'ACTIVE';

SELECT CASE WHEN @active_loan_count = 0 THEN -1 ELSE COUNT (*) END
FROM payments  P
LEFT JOIN user_loan L
ON P.user_loan_id = L.id
WHERE P.user_id = 1
AND L.status = ‘ACTIVE’
```

4.%Gross_15_days_default for all loans disbursed in the past 6 months.

```
SELECT L.id, ((SUM(P.amount)-  MIN(L.amount))/ MIN(L.amount))* 100 as Gross_15_days_default
FROM payments  P
LEFT JOIN user_loan L
ON P.user_loan_id = L.id
WHERE L.status = ‘ACTIVE’
AND DATEDIFF( P.payment_date – P.payment_deadline) > 15
AND TIMESTAMPDIFF(MONTH, L.disbursed_at , now()) <= 6
GROUP BY L.id
```

5.(avg debit amount/avg credit amount) in the sms data from 5 days before to payment_deadline for all "COMPLETED" loans for user_id = 1

```
SELECT SUM( CASE WHEN S.txn_type = ‘DEBIT’ THEN amount ELSE 0 END)/ SUM(( CASE WHEN
S.txn_type = ‘CREDIT’ THEN amount ELSE 0 END)
FROM sms_data  S
LEFT JOIN user_loan L
ON S.user_id = L.user_id
LEFT JOIN payments P
ON L.id = P.user_loan_id
WHERE L.status = ‘COMPLETED’
AND S.user_id = ‘1’
AND DATEDIFF(P.payment_deadline,FROM_UNIXTIME(S.sms_date) ) = 5
```

Q3: List down some features which we can make using the different sources of structured and unstructured data present on user's mobile phone.

Note: Features are the set of independent variables which we can use to train a model. A model determines whether to approve or reject the loan applied by a customer.
For example: From sms data we can make a feature like - Average debit amount over last 30 days for the user. This can be an indicator of the customer's financial capacity.

Possible features :

- Average debit amount
- Monthly credit amount
- Geographical location
- Phone model
- Time of payments
- Demographic mapping
- Apps used (finance apps i.e coinbase, paytm money, groww); time spent on each app