

SQL-Mongo Project – Home Credit Data

BUAN 6320

Evan Jones Boddu

Contents

Data Model	3
Assumptions/Notes About Data Entities and Relationships.....	3
Entity-Relationship Diagram	3
Physical Database	4
Assumptions/Notes About Data Set	4
Screen shot of Physical Database objects.....	4
Data in the Database.....	4
SQL Queries.....	5
Query 1.....	5
Question.....	5
Notes/Comments About SQL Query and Results (Include # of Rows in Result).....	5
Translation	5
Screen Shot of SQL Query and Results.....	5
Query 2.....	6
Question.....	6
Notes/Comments About SQL Query and Results (Include # of Rows in Result).....	6
Translation	6
Screen Shot of SQL Query and Results.....	6
Query 3.....	7
Question.....	7
Notes/Comments About SQL Query and Results (Include # of Rows in Result).....	7
Translation	7
Screen Shot of SQL Query and Results.....	7
Query 4.....	8
Question.....	8
Notes/Comments About SQL Query and Results (Include # of Rows in Result).....	8
Translation	8
Screen Shot of SQL Query and Results.....	8
Query 5.....	9
Question.....	9
Notes/Comments About SQL Query and Results (Include # of Rows in Result).....	9

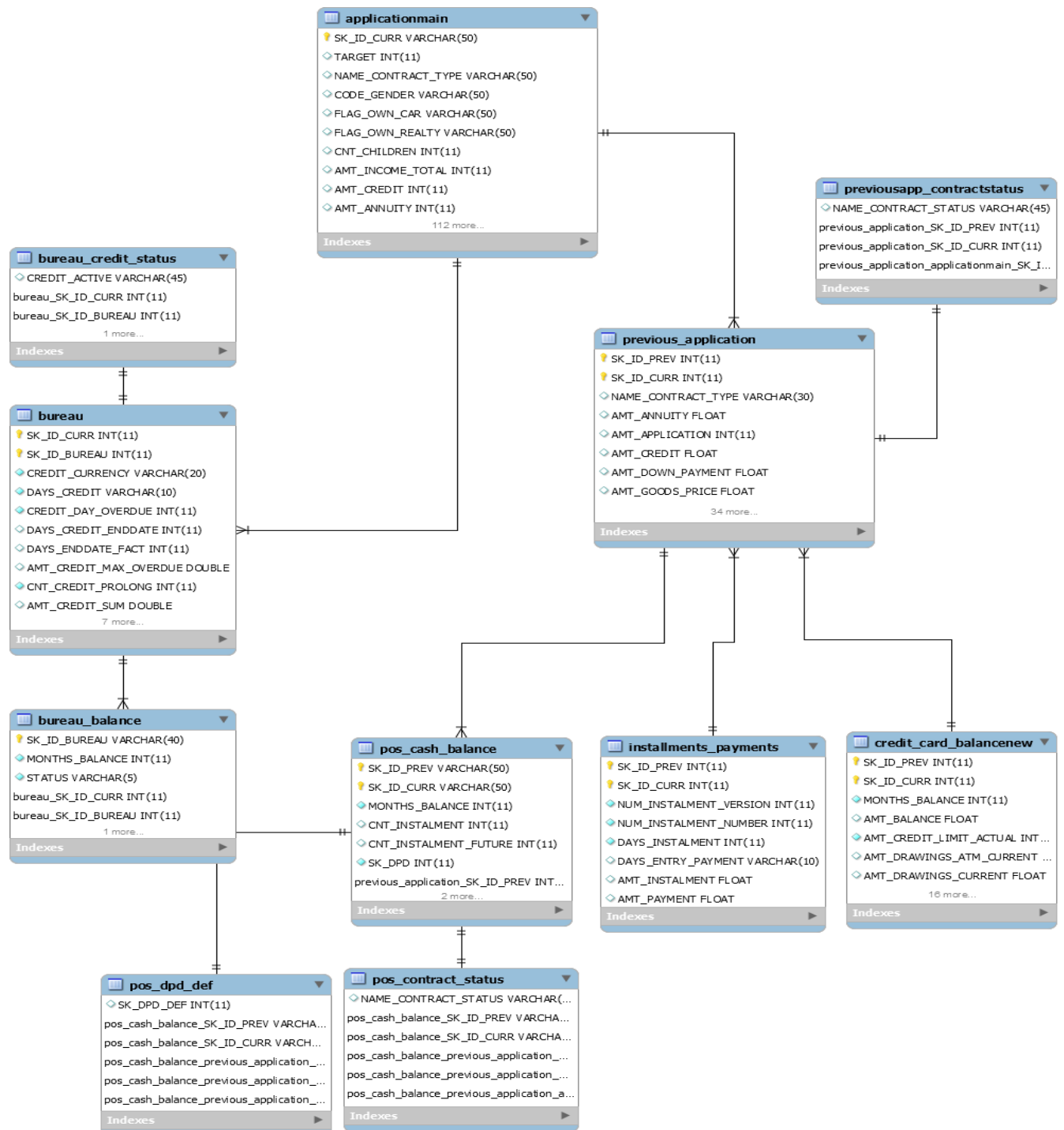
Translation9

Screen Shot of SQL Query and Results.....9

Data Model

Assumptions/Notes About Data Entities and Relationships

Entity-Relationship Diagram



Physical Database

Assumptions/Notes About Data Set

Screen shot of Physical Database objects

Data in the Database

Table Name	Primary Key	Foreign Key	# of Rows in Table
applicationmain	SK_ID_CURR	-	356255
bureau	SK_ID_BUREAU	SK_ID_CURR	1716428
bureau_credit_status	SK_ID_BUREAU	SK_ID_CURR	1048575
bureau_balance	SK_ID_BUREAU	SK_ID_CURR	27299925
previous_application	SK_ID_PREV	SK_ID_CURR	1670214
previousapp_contractstatus	SK_ID_PREV	SK_ID_CURR	1048575
POS_Cash_balance	SK_ID_PREV	SK_ID_CURR	10001358
POS_contract_status	SK_ID_PREV	SK_ID_CURR	1048575
POS_dpd_def	SK_ID_PREV	SK_ID_CURR	1048575
Installments_payments	SK_ID_PREV	SK_ID_CURR	13605401
credit_card_balancenew	SK_ID_PREV	SK_ID_CURR	3840312

SQL Queries

Query 1

Question:

Which occupation types had the highest and lowest number of loans approved?

Notes/Comments About SQL Query and Results (Include # of Rows in Result)

From the occupation type, the highest number of loans approved are for Laborers which are 63841 and the least number of loans approved are for IT Staff which are 607.

Translation

SELECT OCCUPATION_TYPE, and GROUP BY the desc order (highest) and asc order (lowest) number of loans approved.

Screen Shot of SQL Query and Results

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 (SELECT OCCUPATION_TYPE, count(SK_ID_CURR) as COUNT1
2 FROM applicationmain
3 GROUP BY OCCUPATION_TYPE
4 ORDER BY COUNT1 desc
5 limit 2)
6 UNION ALL
7 (SELECT OCCUPATION_TYPE, count(SK_ID_CURR) as COUNT1
8 FROM applicationmain
9 GROUP BY OCCUPATION_TYPE
10 ORDER BY COUNT1 asc
11 limit 1)
```

The Results tab shows the following data:

OCCUPATION_TYPE	COUNT1
IT staff	607
Laborers	63841

The Output tab shows the execution log with the following messages:

#	Time	Action	Message	Duration / Fetch
15	02:40:15	SELECT OCCUPATION_TYPE, count(SK_ID_CURR) as COUNT1 FROM applicationmain GROUP BY OCCU...	1 row(s) returned	2.610 sec / 0.000 sec
16	02:40:24	SELECT OCCUPATION_TYPE, count(SK_ID_CURR) as COUNT1 FROM applicationmain GROUP BY OCCU...	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL s...	0.000 sec
17	02:40:36	SELECT OCCUPATION_TYPE, count(SK_ID_CURR) as COUNT1 FROM applicationmain GROUP BY OCCU...	2 row(s) returned	2.547 sec / 0.000 sec
18	02:41:03	(SELECT OCCUPATION_TYPE, count(SK_ID_CURR) as COUNT1 FROM applicationmain GROUP BY OCC...	3 row(s) returned	5.047 sec / 0.000 sec
19	02:41:23	(SELECT OCCUPATION_TYPE, count(SK_ID_CURR) as COUNT1 FROM applicationmain GROUP BY OCC...	3 row(s) returned	4.953 sec / 0.000 sec
20	02:41:32	(SELECT OCCUPATION_TYPE, count(SK_ID_CURR) as COUNT1 FROM applicationmain GROUP BY OCC...	3 row(s) returned	5.031 sec / 0.000 sec

COUNT of Laborers (the highest) is 63841

COUNT of IT Staff (the lowest) is 607.

Query 2

Question:

What was the average loan amount approved for Revolving Loans?

Notes/Comments About SQL Query and Results (Include # of Rows in Result)

The above code is done using previous_application. Average loan amount is 27767.0604.

Translation

SELECT AVG(AMT_CREDIT) who have been approved for Revolving Loans.

Screen Shot of SQL Query and Results

The screenshot displays the MySQL Workbench interface. The SQL editor shows the following query:

```
1 SELECT COUNT(AMT_CREDIT), SUM(AMT_CREDIT) AS TOTAL, AVG(AMT_CREDIT) AS AVERAGE_LOAN_AMT
2 FROM previous_application
3 WHERE previous_application.NAME_CONTRACT_TYPE LIKE "Revolving Loans"
4 AND previous_application.NAME_CONTRACT_STATUS LIKE "Approved"
```

The Results window shows the following data:

COUNT(AMT_CREDIT)	TOTAL	AVERAGE_LOAN_AMT
97770	22268785500	227767.06044799017

The Output window shows the execution log with the following messages:

#	Time	Action	Message	Duration / Fetch
21	02:44:35	SELECT COUNT(AMT_CREDIT) as AVERAGE_LOAN_AMT FROM previous_application WHERE previous_...	1 row(s) returned	3.640 sec / 0.000 sec
22	02:46:02	SELECT COUNT(AMT_CREDIT) as AVERAGE_LOAN_AMT FROM previous_application WHERE previous_...	1 row(s) returned	7.735 sec / 0.000 sec
23	02:47:33	SELECT COUNT(AMT_CREDIT) as AVERAGE_LOAN_AMT FROM previous_application LIMIT 0, 500	1 row(s) returned	3.704 sec / 0.000 sec
24	02:47:46	SELECT COUNT(AMT_CREDIT) as AVERAGE_LOAN_AMT FROM previous_application WHERE previous_...	1 row(s) returned	3.734 sec / 0.000 sec
25	02:48:05	SELECT COUNT(AMT_CREDIT) as AVERAGE_LOAN_AMT FROM previous_application WHERE previous_...	1 row(s) returned	4.078 sec / 0.000 sec
26	02:49:42	SELECT COUNT(AMT_CREDIT), SUM(AMT_CREDIT) AS TOTAL, AVG(AMT_CREDIT) AS AVERAGE_LOA...	1 row(s) returned	4.453 sec / 0.000 sec

The Average Loan Amount Approved for Revolving loans is 27767.0604. Where the Total is 22268785500 and the count (i.e the number of observations for Revolving loans which are approved) are 97770.

Query 3

Question:

Did applicants with Higher education apply for more Revolving Loans than Cash Loans?

Notes/Comments About SQL Query and Results (Include # of Rows in Result)

No, the applicants who has higher education applied more for cash loans rather than revolving loans. As, the number of applicants who have higher education and applied for cash loans are 77651 and the number of applicants who have higher education and applied for revolving loans are 9728.

Translation

SELECT NAME_EDUCATION_TYPE, NAME_CONTRACT_TYPE, COUNT(NAME_CONTRACT_TYPE) who have higher education and the count of applicants who applied for cash loans and revolving loans.

Screen Shot of SQL Query and Results

The screenshot displays the MySQL Workbench interface. The SQL editor contains the following query:

```
1 SELECT NAME_EDUCATION_TYPE, NAME_CONTRACT_TYPE, count(NAME_CONTRACT_TYPE) AS CASH_OR_REVOLVING
2 FROM applicationmain
3 WHERE NAME_EDUCATION_TYPE LIKE "Higher Education"
4 GROUP BY NAME_CONTRACT_TYPE
```

The Results grid shows the following data:

NAME_EDUCATION_TYPE	NAME_CONTRACT_TYPE	CASH_OR_REVOLVING
Higher education	Cash loans	77651
Higher education	Revolving loans	9728

The Output tab shows the execution log with the following messages:

#	Time	Action	Message	Duration / Fetch
163	17:34:27	SELECT NAME_EDUCATION_TYPE, count(NAME_CONTRACT_TYPE) FROM applicationmain WHERE NA...	1 row(s) returned	1.562 sec / 0.000 sec
164	17:35:07	SELECT NAME_EDUCATION_TYPE, count(NAME_CONTRACT_TYPE), count(NAME_CONTRACT_TYPE) ...	1 row(s) returned	1.500 sec / 0.000 sec
165	17:35:21	SELECT NAME_EDUCATION_TYPE, count(NAME_CONTRACT_TYPE) FROM applicationmain WHERE NA...	1 row(s) returned	1.625 sec / 0.000 sec
166	17:38:24	SELECT NAME_EDUCATION_TYPE, count(NAME_CONTRACT_TYPE) FROM applicationmain WHERE NA...	2 row(s) returned	1.656 sec / 0.000 sec
167	17:38:51	SELECT NAME_EDUCATION_TYPE, count(NAME_CONTRACT_TYPE) AS CASH_OR_REVOLVING FROM...	2 row(s) returned	1.781 sec / 0.000 sec
168	17:39:18	SELECT NAME_EDUCATION_TYPE, NAME_CONTRACT_TYPE, count(NAME_CONTRACT_TYPE) AS CA...	2 row(s) returned	1.734 sec / 0.000 sec

Applications having higher education and cash loans are more than applications having higher education and revolving loans. Cash Loan applicants are 77651, while the Revolving loan applicants are 9728.

Query 4

Question:

What was the average credit utilization percentage for active (not repaid) loan applicants?

Notes/Comments About SQL Query and Results (Include # of Rows in Result)

The average credit utilization percentage for active(unpaid) loan applicants is 38.4133.

Translation

SELECT NAME_CONTRACT_STATUS, AVG(AMT_BALANCE/AMT_CREDIT_LIMIT_ACTAL*100)
where the NAME_CONTRACT_STATUS is active.

Screen Shot of SQL Query and Results

The screenshot displays the MySQL Workbench interface. The SQL editor contains the following query:

```
1 SELECT NAME_CONTRACT_STATUS, AVG(AMT_BALANCE/AMT_CREDIT_LIMIT_ACTAL*100)
2 FROM credit_card_balancenew
3 WHERE NAME_CONTRACT_STATUS = "Active"
```

The Results tab shows the output of the query:

NAME_CONTRACT_STATUS	AVG(AMT_BALANCE/AMT_CREDIT_LIMIT_ACTAL)
Active	38.41329681806962

The Output tab shows the execution log:

#	Time	Action	Message	Duration / Fetch
74	23:51:26	SELECT AMT_BALANCE/AMT_CREDIT_LIMIT_ACTAL*100 as HI, NAME_CONTRACT_STATUS FROM ...	1 row(s) returned	7.250 sec / 0.000 sec
75	23:55:12	SELECT AMT_BALANCE/AMT_CREDIT_LIMIT_ACTAL*100 as AVG_CREDIT_PERCENT, NAME_CON...	1 row(s) returned	7.204 sec / 0.000 sec
76	23:56:15	SELECT NAME_CONTRACT_STATUS, AMT_BALANCE/AMT_CREDIT_LIMIT_ACTAL*100 as AVG_CR...	1 row(s) returned	7.125 sec / 0.000 sec
77	23:56:37	SELECT NAME_CONTRACT_STATUS, AMT_BALANCE/AMT_CREDIT_LIMIT_ACTAL*100 as AVG_CR...	3698436 row(s) returned	0.016 sec / 13.656 sec
78	23:59:21	SELECT NAME_CONTRACT_STATUS, AMT_BALANCE/AMT_CREDIT_LIMIT_ACTAL*100 as AVG_CR...	1 row(s) returned	10.797 sec / 0.000 sec
79	23:59:46	SELECT NAME_CONTRACT_STATUS, AVG(AMT_BALANCE/AMT_CREDIT_LIMIT_ACTAL*100) FROM ...	1 row(s) returned	10.484 sec / 0.000 sec

The Average credit utilization percentage for active loan applicants is 38.4133%

Query 5

Question:

Applicants of which education type had the highest average unused credit?

Notes/Comments About SQL Query and Results (Include # of Rows in Result)

The Applicant of which education type had the highest average unused credit is Academic Degree.

Translation

SELECT NAME_EDUCATION_TYPE where the applicant had the highest average unused credit.

Screen Shot of SQL Query and Results

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 select NAME_EDUCATION_TYPE
2 from (select NAME_EDUCATION_TYPE , amt_credit-amt_annuity as UC from applicationmain
3 group by NAME_EDUCATION_TYPE
4 order by UC desc) B
5 limit 1;
```

The Results pane shows the following result:

NAME_EDUCATION_TYPE
Academic degree

The Output pane shows the following log:

#	Time	Action	Message	Duration / Fetch
111	00:22:11	SELECT previous_application.SK_ID_CURR, applicationmain.SK_ID_CURR, count(previous_application.SK...	Error Code: 2013. Lost connection to MySQL server during query	30.000 sec
112	00:24:24	SELECT NAME_EDUCATION_TYPE, COUNT(SK_ID_CURR) FROM applicationmain GROUP BY NAME_E...	5 row(s) returned	10.250 sec / 0.000 sec
113	00:25:01	SELECT NAME_EDUCATION_TYPE, COUNT(SK_ID_CURR) AS HI FROM applicationmain GROUP BY NA...	5 row(s) returned	11.500 sec / 0.000 sec
114	00:26:13	SELECT NAME_EDUCATION_TYPE, COUNT(SK_ID_CURR) AS HI, AVG(SK_ID_CURR), MAX(SK_ID_CU...	5 row(s) returned	11.312 sec / 0.000 sec
115	00:42:47	SELECT NAME_EDUCATION_TYPE, COUNT(SK_ID_CURR) AS TOTAL, AVG(SK_ID_CURR), MAX(SK_ID...	5 row(s) returned	17.078 sec / 0.000 sec
116	00:51:46	select NAME_EDUCATION_TYPE from (select NAME_EDUCATION_TYPE , amt_credit-amt_annuity as UC f...	1 row(s) returned	10.906 sec / 0.000 sec

The Applicant of which education type had the highest average unused credit is Academic Degree.