*Барышников Илья*

В прикрепленном датасете находятся данные по заявкам клиентов, которые приходили на оформление первого кредита.

Заявкой считается уникальная запись TSApplicationId, поколением является TSApplicationCreatedOn в IST

Необходимо:

1. Изучить датасет и рассказать как поняли данные

2. Воссоздать логику по которой были рассчитаны флаги: is\_completed, is\_approved, is\_disbursed

3. Проанализировать датасет и назвать причины снижения DR в разные дни из-за стадий CR и TR.

1. DR (disbursement rate) - отношение is\_disbursed заявок ко всем

2. CR (completed rate) – отношение is\_completed заявок ко всем

3. TR (take rate) – отношение is\_disbursed к is\_approved

4. Создать процедуру и в ней реализовать инкрементальный забор данных. Представьте, что вы работаете с продовской онлайн таблицей, которая регулярно пополняется. Для целей анализа необходимо забирать только свежий срез и им обогащать конечную витрину для анализа.

5. Предложить бизнес-идеи, которые могли бы улучшить DR.

**!См. также сопровождающий решение файл Task2\_Python\_Baryshnikov.ipynb.**

1. Датасет содержит данные о заявках клиентов, оформлявших первый кредит. Примерные значения полей:

- TSApplicationId – уникальный (зашифрованный?) код (идентификатор) заявки.

- TSApplicationCreatedOn – дата и время создания заявки.

- TSApplicationModifiedOn – дата и время (последнего) изменения данных в заявке.

- TSReasonForLoan – назначение запрашиваемых клиентом средств (кредита).

- TSCustomerType – тип клиента. Так как данные по клиентам, оформившим первый кредит, они все являются новыми (new).

- DisbursementType – тип выплаты (автоматический, ручной – как была осуществлена выплата).

- TSApplicationStatus – статус заявки (отказ, регистрация, выберите сумму и т.д.)

- Product - ?кредитный продукт (PDL и IL; PDL (Pay Day Loans – «займы до зарплаты) – краткосрочные займы на небольшую сумму, IL (Instalment Loan) характеризуются большей продолжительностью, установленным количеством запланированных платежей и большими сроками)

- DecisionId – код/идентификатор решения (по кредиту)

- TSAppRejectionReason – причина отказа в выдаче кредита

- DecisionCreatedOn – дата и время (создания) решения по кредиту

- DecisionModifiedOn – дата и время (последнего) изменения по кредиту

- TSAgreementId – уникальный идентификатор соглашения с клиентом

- TSAgreementCreatedOn – дата и время (создания) соглашения

- TSAgreementModifiedOn – дата и время (последнего) изменения в соглашении

- TSAgreementStatus – статус соглашения (активно, перевыплата (Overpayment), отмена, готово к выплате)

- ContactId – уникальный id контакта (клиента)

- Is\_completed – заполнена ли заявка (1 – да, 0 – нет)

- is\_approved – одобрена ли заявка (1 – да, 0 – нет)

- is\_disbursed – произведена ли выплата (1 – да, 0 – нет)

- duplicatedTSApplication – информация о дубликате строки (в датасете есть несколько строчек с одним и тем же TSApplicationID, всего 61 заявка встречается дважды; различаются по этому полю (TSApplicationID и duplicatedTSApplication вместе уникально идентифицируют строчку)), принимает значения 1 или 2

- is\_post\_decision\_completed - ?вероятнее всего, флаг, который показывает, что клиент осведомлен о решении (и прошёл определенный клиентский путь, как правило, это success\_screen и agreement\_for\_personal\_loan)

- BanksListName - ?указанный клиентом банк (при вводе реквизитов)

- rep\_date – предположительно, дата обработки заявки (можно увидеть, что иногда там указан следующий день, если заявка создана поздно предыдущим днём)

Шаги 1, 2, 3 показывают «клиентский путь» (скорее всего, последние страницы и коды ответа с них):

- step\_name\_1 - название шага 1

- status\_code\_1 – статус/код ответа со страницы (200 – успешно, 400 – ошибка Bad Request (ошибка на стороне пользователя), 500 – внутренняя ошибка сервера, 404 – Not Found (нет данных по запросу клиента), 401 – ошибка Unathorized (неверная авторизация, попытка получить доступ к странице для авторизованного пользователя и т.п.))

- step\_name\_2 – название шага 2

- status\_code\_2 – статус/код ответа со страницы

- step\_name\_3 – название шага 3

- status\_code\_3 - статус/код ответа со страницы

Важное значение имеет поле duplicatedTSApplication, которое указывает на то, что строчка с данным TSApplicationID не единственная. Как было замечено, происходит дублирование (добавление новой строки с таким же TSApplicationID) при изменении данных о решении по заявке (в т.ч. отличается DecisionID, DecisionCreatedOn и т.п.). **Предположительно, наиболее поздние (актуальные) данные о решении по заявке хранятся в строке с duplicatedTSApplication = 1, а в ранних записях duplicatedTSApplication присваивается значение 2. Если это предположение об организации данных верно, это усложняет логику процедуры инкрементального сбора данных в пункте 4, которую я постараюсь предусмотреть.**

1. Флаги is\_completed, is\_approved, is\_disbursed отвечают за статус, в котором находится заявка (заполнено, одобрено, выплачено – очевидно, самым ранним является статус «заполнено», а завершающим – «выплачено». Очевидно, что денежные средства не могут быть «выплачены» (is\_disbursed = 1) без заполнения заявки (is\_completed = 1) и одобрения (is\_approved = 1)). Существует логика/связь в данных между значением флагов и шагами, которые прошёл пользователь (step\_name\_1...3).

Флаг is\_completed имеет значение «0», когда пользователь не прошёл/остановился на ранней стадии (регистрация, не до конца ввёл данные (step\_name = bank\_details/aadhaar)) или попал на BlackListScreen после регистрации или шага aadhaar/aadhaar\_kyc (ввод идентификационного номера (Индия)).

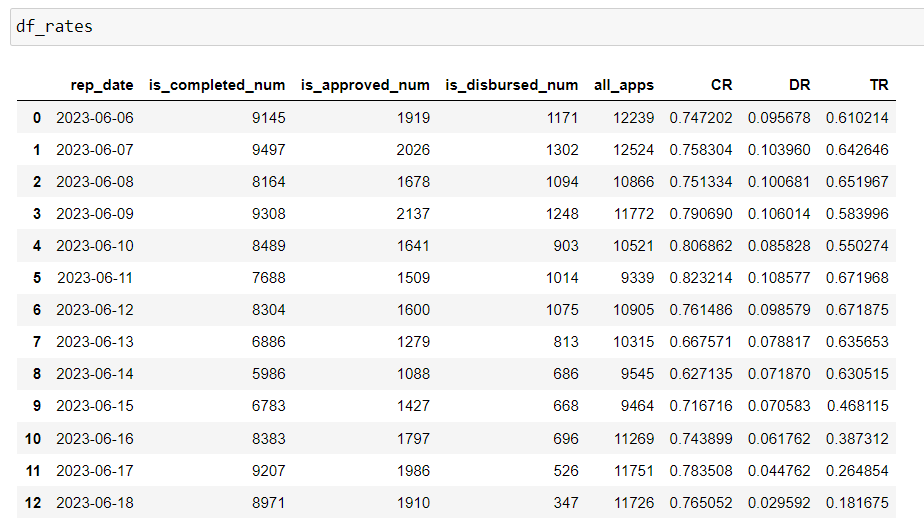
Флаг is\_completed имеет значение «1», как правило, когда пользователь ввёл свои данные (прошёл шаги bank\_details).

Флаг is\_approved может иметь значение «0» (при заполненной заявке), когда клиент после введения банковских данных был идентифицирован как клиент из чёрного списка (он попадает на BlackScreenList) или остановился на странице авторизации (auth\_otp).

Флаг is\_approved («одобрено») со значением «1» может появиться только после заполнения данных (шагов bank\_details (с кодом 200), однако пользователь может ещё не пройти дальше, как и в случае с is\_approved=0). Пользователь может находиться на следующих далее шагах аутентификации (auth\_otp), выбора условий кредита/выбора кредитного продукта (choose\_loan\_pdl, choose\_loan\_pdl\_details), быть на стадии заключения соглашения (agreement\_for\_personal\_loan), пройти success\_screen (но ещё фактически не получить выплату).

Флаг is\_disbursed имеет значение «1», когда пользователь фактически получил выплату. При этом он прошёл через страничку о том, что займ успешно предоставлен (step\_name = success\_screen; также встречается agreement\_personal\_loan)

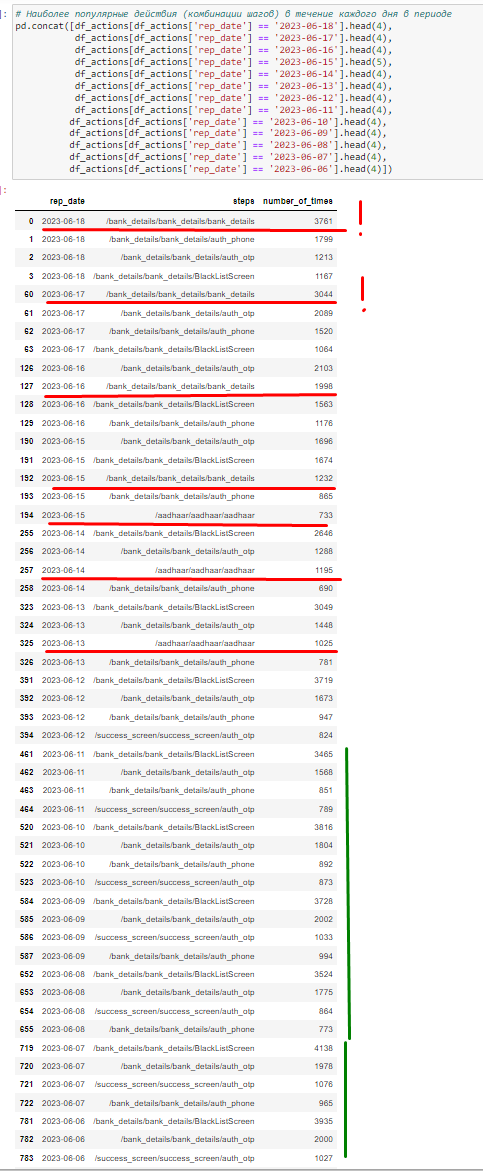
1. Сгруппируем данные по дням (удобнее использовать для этого rep\_date, см. это поле в п.1) и рассчитаем число заполненных, одобренных заявок, число выплат, а также следующие метрики:
   * DR (disbursement rate) - отношение is\_disbursed заявок ко всем
   * CR (completed rate) – отношение is\_completed заявок ко всем
   * TR (take rate) – отношение is\_disbursed к is\_approved



По полученным данным можно увидеть, что с 13 июня наблюдается снижение DR, однако это снижение в разные дни сопровождается разным поведением CR и TR:

1. 13-14.06.2023 (отчасти можно отнести и 15.06) – самые низкие значения CR в рассматриваемом периоде, после чего показатель восстанавливается. Гораздо меньшее число пользователей прошли стадию заполнения заявки.
2. 15-18.06.2023 – глубокое падение TR. Резкое падение кол-ва выплаченных кредитов.

Наиболее популярные комбинации шагов пользователей показывают, что 13-14.06 наблюдался резкий рост количества заявок/пользователей, остановившихся на этапе идентификации aadhaar. После наблюдается резкий рост «остановок» на вводе банковских данных (аналогично причинам падении конверсии в первом тестовом задании). Можно предположить, что проблемы на соответствующих страницах, проблемы с подключением/обменом данных с внешними ресурсами могли стать причиной падения количества выплат.



Проведение группировки по наиболее частым причинам отказа в выдаче кредита не выявляет резкого изменения паттерна в период 13-18.06.2023 (разве что меньшее количество отказов от скоринговой модели 14.06). Однако для 5 пункта задания отметим стабильно высокое количество отказов из-за возраста клиента и большое количество Bureau Ping Error (возможно, это подключение к какому-то внешнему источнику данных).



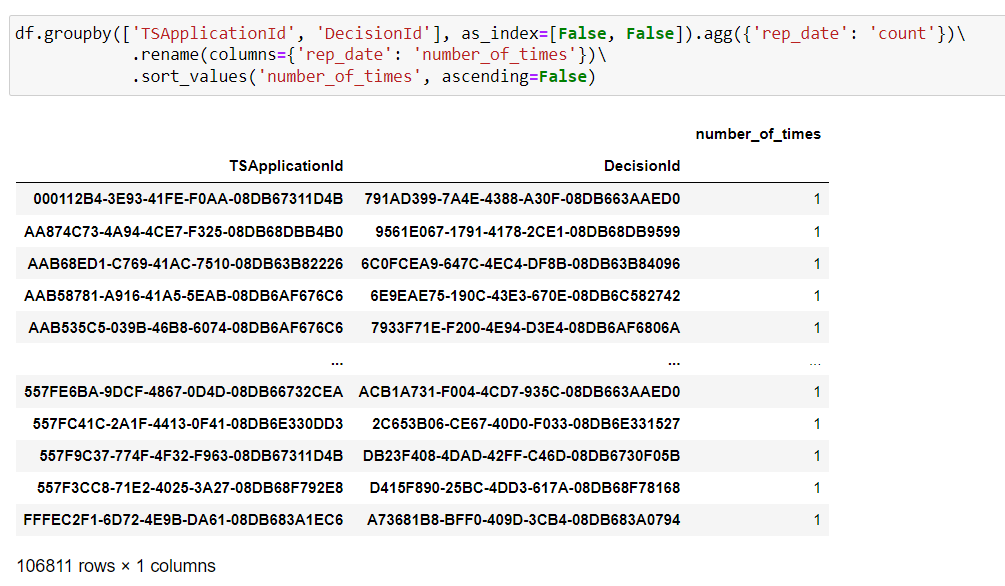
Группировка по TSApplicationStatus подтверждает резкий взлёт заявок со статусом aadhaar verification 13-14.06, а также Choose loan amount в последние дни периода.



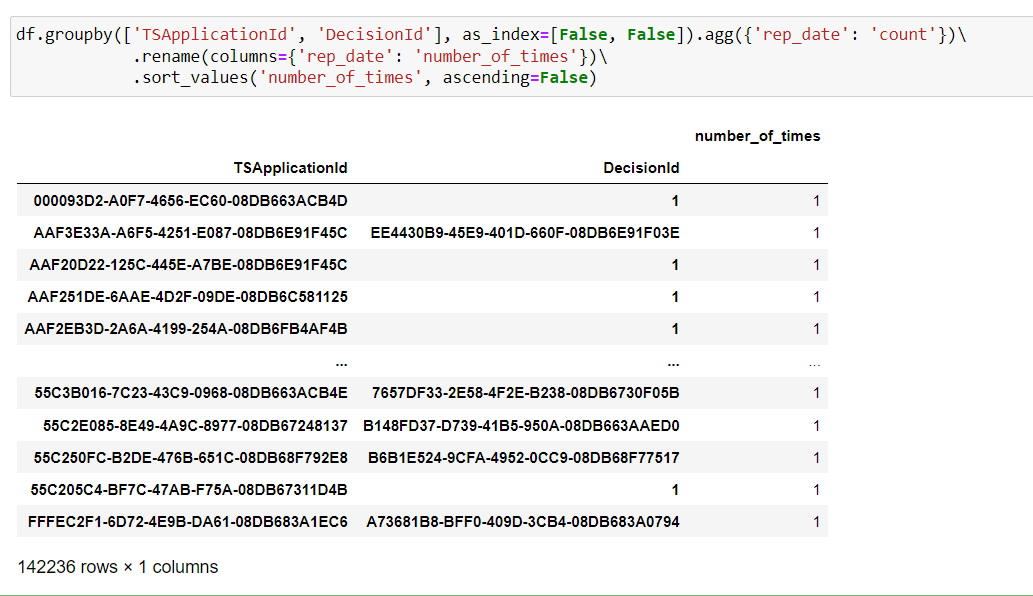
Наиболее вероятными причинами падения DR в разные дни стали проблемы со страницами идентификации aadhaar и ввода банковских данных.

4. Хранимая процедура доступна отдельно в файле incremental\_refresh\_Baryshnikov.sql.

Ниже мы можем убедиться, что TSApplicationId и DecisionId уникально идентифицируют строку в таблице (скорее всего, предположение, что дублирующие строчки добавляются только для уточнения информации о решении по заявке подтверждается).



Доп.проверка: здесь NULL в DecisionId заменены на «1» - также нет дубликатов по двум полям.



**Примечание: в SQL Server я создал таблицу-источник с уникальным ключом по полям TSApplicationId и duplicatedTSApplication (использовать DecisionId нельзя,т.к. поле может принимать значение NULL).**

**Исходя из ранее полученной логики, предполагаем, что в старых записях по заявкам (если есть дубликат по TSApplicationId) меняется только поле duplicatedTSApplication (в общем случае можно делать апдейт всех полей, если таблица небольшая). Непосредственно за добавление свежего среза (новых строк) отвечает часть «**WHEN NOT MATCHED THEN INSERT…»**. С учётом усложненной логики (более точный, оптимизированный и универсальный код можно было бы сделать при однозначном понимании устройства данных и отсутствии проблем с дубликатами) процедура инкрементального сбора данных будет выглядеть следующим образом:**

CREATE PROCEDURE incremental\_refresh

AS

BEGIN

SET NOCOUNT ON;

--если нет целевой таблицы, создадим её

IF NOT EXISTS (SELECT \* FROM sys.tables t WHERE t.name = 'ntb\_funnel\_target')

CREATE TABLE [dbo].[ntb\_funnel\_target](

[TSApplicationId] [nvarchar](50) NOT NULL,

[TSApplicationCreatedOn] [datetime2](7) NULL,

[TSApplicationModifiedOn] [datetime2](7) NULL,

[TSReasonForLoan] [nvarchar](50) NULL,

[TSCustomerType] [nvarchar](50) NULL,

[DisbursementType] [nvarchar](50) NULL,

[TSApplicationStatus] [nvarchar](50) NULL,

[Product] [nvarchar](50) NULL,

[DecisionId] [nvarchar](50) NULL,

[TSAppRejectionReason] [nvarchar](100) NULL,

[DecisionCreatedOn] [nvarchar](50) NULL,

[DecisionModifiedOn] [nvarchar](50) NULL,

[TSAgreementId] [nvarchar](50) NULL,

[TSAgreementCreatedOn] [nvarchar](50) NULL,

[TSAgreementModifiedOn] [nvarchar](50) NULL,

[TSAgreementStatus] [nvarchar](50) NULL,

[ContactId] [nvarchar](50) NULL,

[is\_completed] [bit] NULL,

[is\_approved] [bit] NULL,

[is\_disbursed] [bit] NULL,

[duplicatedTSApplication] [tinyint] NOT NULL,

[is\_post\_decision\_completed] [bit] NULL,

[BanksListName] [nvarchar](max) NULL,

[rep\_date] [date] NULL,

[step\_name\_1] [nvarchar](50) NULL,

[status\_code\_1] [nvarchar](50) NULL,

[step\_name\_2] [nvarchar](50) NULL,

[status\_code\_2] [nvarchar](50) NULL,

[step\_name\_3] [nvarchar](50) NULL,

[status\_code\_3] [nvarchar](50) NULL,

CONSTRAINT [PK\_ntb\_funnel\_target] PRIMARY KEY CLUSTERED

(

[TSApplicationId] ASC,

[duplicatedTSApplication] ASC

))

--ntb\_funnel - таблица-источник (s - source), ntb\_funnel\_target - целевая таблица

MERGE ntb\_funnel\_target AS t

USING (SELECT \* FROM ntb\_funnel) AS s

ON t.TSApplicationId = s.TSApplicationId AND t.DecisionId = s.DecisionId

--Если совпадают поля, обновляем флаг дубликата (согласно полученной в ходе анализа логике меняется лишь этот флаг, данные о новом решении - в новой строке с новым DecisionId)

WHEN MATCHED AND t.duplicatedTSApplication != s.duplicatedTSApplication THEN UPDATE SET t.duplicatedTSApplication = s.duplicatedTSApplication

--Если нет совпадающих (по условиям) строк, добавляем новые из источника

WHEN NOT MATCHED THEN INSERT VALUES (s.TSApplicationId,

s.TSApplicationCreatedOn,

s.TSApplicationModifiedOn,

s.TSReasonForLoan,

s.TSCustomerType,

s.DisbursementType,

s.TSApplicationStatus,

s.Product,

s.DecisionId,

s.TSAppRejectionReason,

s.DecisionCreatedOn,

s.DecisionModifiedOn,

s.TSAgreementId,

s.TSAgreementCreatedOn,

s.TSAgreementModifiedOn,

s.TSAgreementStatus,

s.ContactId,

s.is\_completed,

s.is\_approved,

s.is\_disbursed,

s.duplicatedTSApplication,

s.is\_post\_decision\_completed,

s.BanksListName,

s.rep\_date,

s.step\_name\_1,

s.status\_code\_1,

s.step\_name\_2,

s.status\_code\_2,

s.step\_name\_3,

s.status\_code\_3)

--Вывести изменения

OUTPUT $action, inserted.\*, deleted.\*;

END

GO

--выполнить процедуру

exec incremental\_refresh

5. На основе проведённого анализа среди общих направлений, позволяющих повысить DR (отношение количества выплаченных клиентам кредитов к общему числу заявок), можно выделить следующие:

- упрощение и либерализация скоринговой модели (самая популярная причина отказов - Scoring Model Reject). Возможна выборочная либерализация для определённого сегмента пользователей (на основе данных о доходе, возрасте, работе и т.п.).

- максимальное упрощение интерфейса приложения (минимизация, насколько возможно, количества вводимых полей. В качестве отвлечённого примера можно привести отсутствие необходимости в графе «кем выдан» при наличии графы «код подразделения» (если бы вводились данные паспорта РФ)).

- наличие альтернативных методов идентификации, проверки данных (для защиты от сбоев в работе внешних сервисов/интеграций, например, при сверке aadhaar).

- проверку возраста, возможно, стоит ввести максимально рано в интерфейсе (возможно, до формальной «фиксации» заявки).

- решить проблему Bureau Ping Error (?подключение к Bureau?)