Презентация проекта Мининой Е.Д. на тему:

«Формирование отчетности по форме 101: Данные оборотной ведомости по счетам бухгалтерского учёта»

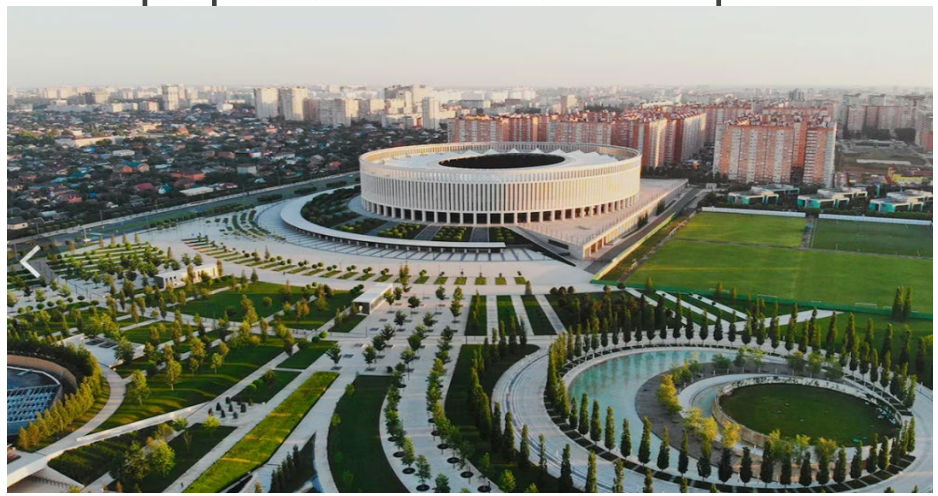
В рамках проектного задания
УЦ Неофлекс 2025 аналитика nr

Этапы презентации

- Немного о себе
- Цель проекта
- Сбор и работа с требованиями
- Подготовка к прототипированию
- Прототипирование
- Проверка прототипа и визуализация данных
- Итог
- Общие впечатления

Немного о себе

Проживаю в г. Краснодар, мне 32 года, бухгалтер по образованию и профессии. По окончании университета профессиональный опыт работы получала в Москве.



Работала бухгалтером в:

- аутсорсинговой,
- строительной,
- нефте-газовой,
- логистической организациях.

Имею опыт ведения бухгалтерского учета и управленческой отчетности.

Хочу развиваться дальше, вижу для себя перспективы развития в аналитике. Начала свой путь в обучении на аналитика в декабре 2024г.

Мои преимущества:

- ✓ понимание бухгалтерского учета,
- ✓ умение пользоваться законодательной базой,
- ✓ базовые знания SQL и желание обучаться дальше.

Цель проекта:

- ✓ Настроить репликацию данных из БД АБС в Хранилище данных;
- ✓ Построить аналитическую витрину «Форма 101: Данные оборотной ведомости по счетам бухгалтерского учёта»;
- ✓ Снижение рисков штрафов и иных санкций, возможных из-за предоставления некорректной отчетности или ее непредоставления;
- ✓ Организация процесса подготовки аналитической отчетности для финансового мониторинга банка;
- ✓ Существенное снижение трудоемкости формирования документов/отчетов.

Сбор и работа с требованиями

Сбор требований производился с помощью следующих методов:	
Use case	На основе которых были детально описаны взаимодействие между пользователем и системой, по следующим сценариям: -Создание отчета; -Выгрузка сформированного отчета.
User story	Пользовательские истории были сформированы в ходе целевого опроса представителей ключевых ролей: • Учредитель - Ценность для бизнеса • Руководитель -> Эффективность процессов • Бухгалтер/Экономист -> Корректность данных и отчетности • Системный администратор -> Техническая реализуемость и безопасность.
Чек-лист вопросов заказчику	Для обеспечения полноты и согласованности требований, был применен метод структурированного интервью, на основе детального чек-листа.

Функциональные требования, которыми должна обладать система:

- Формирование отчета должно производиться по любому запросу пользователя.
- Отчет должен формироваться по запросу, за любой выбираемый период.
- Система должна включать процесс авторизации, для предоставления уникальных прав пользователю.
- Интерфейс программы должен быть прост и понятен.
- Система должна интегрироваться со всеми необходимыми системами банка.
- Система должна предоставлять выбор, места выгрузки, при сохранении отчёта вне системы.
- Система должна выводить абсолютно точные и полные данные в отчет.
- Система должна сохранять отчет, за период, который уже был сдан по требованию ЦБ РФ.
- Отчет из системы должен выгружаться в формате Excel.

Нефункциональные требования:

1.Требования к производительности:

- Система должна иметь высокую скорость отклика на запросы при стандартной работе и пиковых нагрузках: до 5 секунд
- Быстро обрабатывать большие объёмы данных и сложные запросы, обеспечивая при этом стабильность и доступность данных
- Поддерживать одновременную работу всех существующих пользователей-бесперебойно
- Быть легкой в установке и настройке

2. Конкурентная работа:

В момент ввода в эксплуатацию в системе будет работать порядка 30 пользователей.

3. Требования к объемам данных:

В имеющейся АБС предоставлен архив записей за 13 лет, размером более 1 млн., строк. Исходя из этого, объем СУБД должен составлять 1ТБ.

4.Требования по защите информации

Требования к создаваемой системе АС выдвигаются на основании действующих в РФ нормативных, правовых, руководящих документов и национальных стандартов по информационной безопасности. АС не является государственной информационной системой.

5.Требования к мерам защиты информации:

✓ Идентификация и аутентификация субъектов доступа и объектов доступа;
✓ Управление доступом субъектов доступа к объектам доступа;
✓ Ограничение программной среды;
✓ Защита машинных носителей информации;
✓ Регистрация событий безопасности;
✓ Антивирусная защита;
✓ Обнаружение (предотвращение) вторжений;
✓ Контроль (анализ) защищенности информации;
✓ Целостность информационной системы и информации;
✓ Доступность информации;
✓ Защита среды виртуализации;
✓ Защита технических средств;
✓ Защита информационной системы, ее средств, систем связи и передачи данных;
✓ Защита информационной системы от атак, направленных на отказ в обслуживании.

Подготовка к прототипированию, предварительный анализ данных

На этом этапе, для упрощения создания будущего прототипа, его можно разбить на отдельные простые запросы, которые решают следующие задачи:

- Вывести список счетов, актуальных в заданном периоде дат.
- Определить даты последнего движения по счету, по состоянию на определенную дату.
- Определить исходящий остаток по счету на определенную дату, в рублях.
- Определить кредитовый оборот по счету в заданном периоде дат, в рублях.
- Определение дебетового оборота по счету, в заданном периоде дат, в рублях.

SQL-запросы представлены в Приложении 1.

На этапе «Прототипирование» будет выполнено объединение ранее выполненных запросов.

Прототипирование

Этап 1:Определим временные границы остатков

- По каждому счету найдем последние даты остатков на начало периода

```
last_dates_incoming as (
    SELECT account, MAX(date_carry) as max_date_carry
    FROM bank.restdate_dbt
    where date_carry < '2021-05-01'
    GROUP BY account
```

- По каждому счету найдем последние даты остатков на конец периода

```
last_dates_outgoing as (
    SELECT account, MAX(date_carry) as max_date_carry
    FROM bank.restdate_dbt
    where date_carry < '2021-06-01'
    GROUP BY account
)
```

Этап 2: Фильтрация активных счетов (filtered_accounts)

На этом этапе находим:

- Активные или Пассивные счета
- Счета открытые до или на конец периода (31.05.2021г.)
- Счета не закрытые либо закрытые после начала периода

```
filtered_accounts as (
    SELECT
        account,
        balance,
        chapter,
        kind_Account,
        Code_Currency
    FROM bank.account_dbt
    where (Kind_Account = 'А' or Kind_Account = 'П')
      and Open_Date <= '2021-05-31'
      and (Close_Date IS NULL or Close_Date >= '2021-05-01')
      and account in (SELECT distinct ac.account
    FROM bank.account_dbt ac
    where ac.Kind_Account in ('А', 'П')
                and ac.Open_Date <= '2021-05-31'
                and (Close_Date IS NULL or Close_Date >= '2021-05-01')
```

- Счета по которым были движения в мае 2021г.:

```
and EXISTS (SELECT 1
            FROM bank.arhdoc_dbt ar
            where (ac.account = ar.Real_Payer or ac.account = ar.Real_Receiver)
            and ar.chapter=ac.chapter
            and ar.date_carry BETWEEN '2021-05-01' and '2021-05-31')
```

- Счета с ненулевыми остатками на 31.05.2021г.

```
union
        select distinct tt.account
        from (SELECT account, max(date_carry) as carry, chapter
                FROM bank.restdate_dbt
                where date_carry<='2021-05-31'
                group by account, chapter) as tt
        join bank.restdate_dbt as r on tt.account=r.account
        and tt.carry =r.date_carry and tt.chapter=r.chapter
        where r.rest != 0 and exists(select 1 from bank.account_dbt as ac
        where ac.account=r.account and ac.Kind_Account in ('А', 'П') and ac.chapter=r.chapter
```

Этап 3: Расчет входящих остатков (incoming_balances)

На этом этапе находим входящие остатки по счетам, принимая во внимание, что в форме 101, обязательны к выполнению следующие условия:

- суммы должны быть положительными,
- участвует банковское округление,
- отчет составляется в тыс.руб.

```
incoming_balances as (SELECT
                        ac.balance,
                        ac.chapter,
                        kind_Account,
        ROUND(ABS(SUM(case WHEN re.account LIKE '_____810%' THEN rest ELSE 0 END)/1000)) as VR,
        ROUND(ABS(SUM(case WHEN re.account NOT LIKE '_____810%' THEN rest ELSE 0 END)/1000)) as VC,
        ROUND(ABS(SUM(case WHEN re.account LIKE '_____098%' THEN rest ELSE 0 END)/1000)) as VG,
        ROUND(ABS(SUM(case WHEN re.account LIKE '_____099%' or
        re.account LIKE '_____076%' or
        re.account LIKE '_____033%' THEN rest ELSE 0 END)/1000)) as VM,
        ROUND(ABS(SUM(rest)/1000)) as VITG
    FROM filtered_accounts ac
    JOIN bank.restdate_dbt re on re.account = ac.account
                            and re.Chapter = ac.chapter
                            and re.Code_Currency = ac.Code_Currency
    JOIN last_dates_incoming ld on re.account = ld.account and re.date_carry = ld.max_date_carry
    where re.Code_Currency = 0
    GROUP BY ac.balance, ac.chapter, ac.kind_Account
```

Где: 098- золото, 099- серебро, 076-платина, 033- палладий

Этап 4: Расчет оборотов за май 2021г.(turnovers)

На этом этапе выделяем:

- сумму дебетовых оборотов

```
turnovers as ( SELECT
                ac.balance,
                ac.chapter,
                ac.kind_Account,
                ac.Code_Currency,

    --Сумма дебетовых оборотов

    SUM(case WHEN ar.Real_Payer = ac.account and ac.account LIKE '_____810%' THEN
        case WHEN ar.code_currency = ac.Code_Currency THEN ar.sum ELSE 0 END
    ELSE 0 END) as sum_ora,

    SUM(case WHEN ar.Real_Payer = ac.account and ac.account NOT LIKE '_____810%' THEN
        case WHEN ar.code_currency = ac.Code_Currency THEN ar.sum ELSE 0 END
    ELSE 0 END) as sum_oca,

    SUM(case WHEN ar.Real_Payer = ac.account and ac.account LIKE '_____098%' THEN
        case WHEN ar.code_currency = ac.Code_Currency THEN ar.sum ELSE 0 END
    ELSE 0 END) as sum_oga,

    SUM(case WHEN ar.Real_Payer = ac.account and (ac.account LIKE '_____099%' or
    ac.account LIKE '_____076%' or ac.account LIKE '_____033%') THEN
        case WHEN ar.code_currency = ac.Code_Currency THEN ar.sum ELSE 0 END
    ELSE 0 END) as sum_oma,
```

- сумму кредитовых оборотов

```
    SUM(case WHEN ar.Real_Receiver = ac.account and ac.account LIKE '_____810%' THEN
        case WHEN ar.code_currency = ac.Code_Currency THEN ar.sum ELSE 0 END
    ELSE 0 END) as sum_orp,

    SUM(case WHEN ar.Real_Receiver = ac.account and ac.account NOT LIKE '_____810%' THEN
        case WHEN ar.code_currency = ac.Code_Currency THEN ar.sum ELSE 0 END
    ELSE 0 END) as sum_ocp,

    SUM(case WHEN ar.Real_Receiver = ac.account and ac.account LIKE '_____098%' THEN
        case WHEN ar.code_currency = ac.Code_Currency THEN ar.sum ELSE 0 END
    ELSE 0 END) as sum_ogp,

    SUM(case WHEN ar.Real_Receiver = ac.account and (ac.account LIKE '_____099%' or
    ac.account LIKE '_____076%' or ac.account LIKE '_____033%') THEN
        case WHEN ar.code_currency = ac.Code_Currency THEN ar.sum ELSE 0 END
    ELSE 0 END) as sum_omp
FROM filtered_accounts ac
JOIN bank.arhdoc_dbt ar on (ac.account = ar.Real_Payer or ac.account = ar.Real_Receiver)
where ar.date_carry BETWEEN '2021-05-01' and '2021-05-31'
 and ac.code_currency = 0
 and ar.Real_Payer != ar.Real_Receiver
 and ar.code_currency = ac.Code_Currency
GROUP BY ac.balance, ac.chapter, ac.kind_Account, ac.Code_Currency
),
```

Этап 5: Расчет исходящих остатков

- Производим аналогично расчёту входящих остатков

```
outgoing_balances as (
    SELECT
        ac.balance,
        ac.chapter,
        kind_Account,
        ROUND(ABS(SUM(case WHEN re.account LIKE '_____810%' THEN rest ELSE 0 END)/1000)) as IR,
        ROUND(ABS(SUM(case WHEN re.account NOT LIKE '_____810%' THEN rest ELSE 0 END)/1000)) as IC,
        ROUND(ABS(SUM(case WHEN re.account LIKE '_____098%' THEN rest ELSE 0 END)/1000)) as IG,
        ROUND(ABS(SUM(case WHEN re.account LIKE '_____099%' or
 re.account LIKE '_____076%' or
 re.account LIKE '_____033%' THEN rest ELSE 0 END)/1000)) as IM,
        ROUND(ABS(SUM(rest)/1000)) as IITG
    FROM filtered_accounts ac
    JOIN bank.restdate_dbt re on re.account = ac.account
                             and re.Chapter = ac.chapter
                             and re.Code_Currency = ac.Code_Currency
    JOIN last_dates_outgoing ld on re.account = ld.account and re.date_carry = ld.max_date_carry
    where re.Code_Currency = 0
    GROUP BY ac.balance, ac.chapter, ac.kind_Account
```

Этап 6: Сборка финального запроса, включая все выше описанные условия

- Производим замену наименований глав
- Производим замену наименований активных и пассивных счетов
- Выделаем дату, на которую составлен отчет

```
SELECT DISTINCT on (ac.balance)
    case
        WHEN ac.chapter = 1 THEN 'А.Балансовые счета'
        WHEN ac.chapter = 2 THEN 'Б.Счета доверительного управления'
        WHEN ac.chapter = 3 THEN 'В.Внебалансовые счета'
        WHEN ac.chapter = 4 THEN 'Г.Срочные операции'
        ELSE 'Неизвестный раздел'
    END as PLAN,
    ac.balance as NUM_SC,
    case
        WHEN ac.kind_Account = 'А' THEN 'а'
        WHEN ac.kind_Account = 'П' THEN 'р'
        ELSE 'н'
    END as A_P,
    '2021-06-01' as DT,
```

- Применим COALESCE для всех суммарных значений, чтобы заменить все значения NULL на 0. Для корректного выведение отчета и итоговых значений.

```
-- Входящие остатки
COALESCE(ib.VR, 0) as VR,
COALESCE(ib.VC, 0) as VC,
COALESCE(ib.VG, 0) as VG,
COALESCE(ib.VM, 0) as VM,
COALESCE(ib.VITG, 0) as VITG,
-- Обороты
COALESCE(ROUND(ABS(t.sum_ora)/1000), 0) as ORA,
COALESCE(ROUND(ABS(t.sum_oca)/1000), 0) as OCA,
COALESCE(ROUND(ABS(t.sum_oga)/1000), 0) as OGA,
COALESCE(ROUND(ABS(t.sum_oma)/1000), 0) as OMA,
COALESCE(ROUND(ABS(t.sum_ora + t.sum_oca + t.sum_oga + t.sum_oma)/1000), 0) as OITGA,
COALESCE(ROUND(ABS(t.sum_orp)/1000), 0) as ORP,
COALESCE(ROUND(ABS(t.sum_ocp)/1000), 0) as OCP,
COALESCE(ROUND(ABS(t.sum_ogp)/1000), 0) as OGP,
COALESCE(ROUND(ABS(t.sum_omp)/1000), 0) as OMP,
COALESCE(ROUND(ABS(t.sum_orp + t.sum_ocp + t.sum_ogp + t.sum_omp)/1000), 0) as OITGP,
-- Исходящие остатки
COALESCE(ob.IR, 0) as IR,
COALESCE(ob.IC, 0) as IC,
COALESCE(ob.IG, 0) as IG,
COALESCE(ob.IM, 0) as IM,
COALESCE(ob.IITG, 0) as IITG
```

- В финальной части запроса производим соединение через LEFT JOIN следующих таблиц поэтапно:

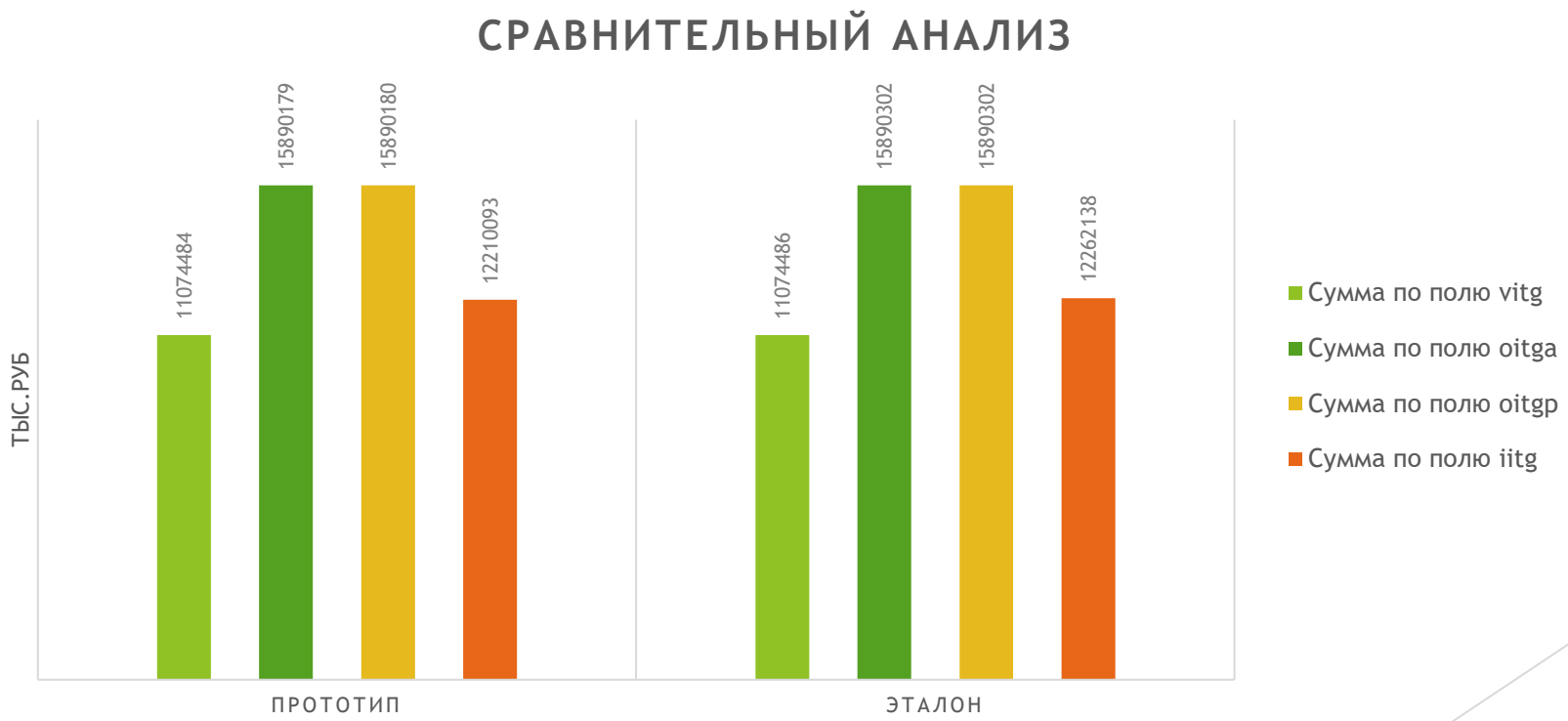
1. filtered_accounts (отфильтрованные счета)
2. incoming_balances (входящие остатки)
3. turnovers (обороты)
4. outgoing_balances (исходящие остатки)

```
FROM filtered_accounts ac
LEFT JOIN incoming_balances ib on ib.balance = ac.balance
                    and ib.chapter = ac.chapter
                    and ib.kind_Account = ac.kind_Account
LEFT JOIN turnovers t on ac.balance = t.balance
                    and ac.chapter = t.chapter
                    and ac.kind_Account = t.kind_Account
LEFT JOIN outgoing_balances ob on ac.balance = ob.balance
                              and ac.chapter = ob.chapter
                              and ac.kind_Account = ob.kind_Account
ORDER BY ac.balance, ac.chapter, ac.kind_Account;
```

Проверка прототипа и визуализация данных

В ходе сравнительного анализа прототипа и эталона, по итоговым значениям, были выявлены суммарные отклонения.

Источник	Сумма по полю vitg	Сумма по полю oitga	Сумма по полю oitgp	Сумма по полю iitg
Прототип	11074484	15890179	15890180	12210093
Эталон	11074486	15890302	15890302	12262138
Отклонение	-2	-123	-122	-52045



Исходя из выявленных отклонений, произведём более детальный анализ по разделам.

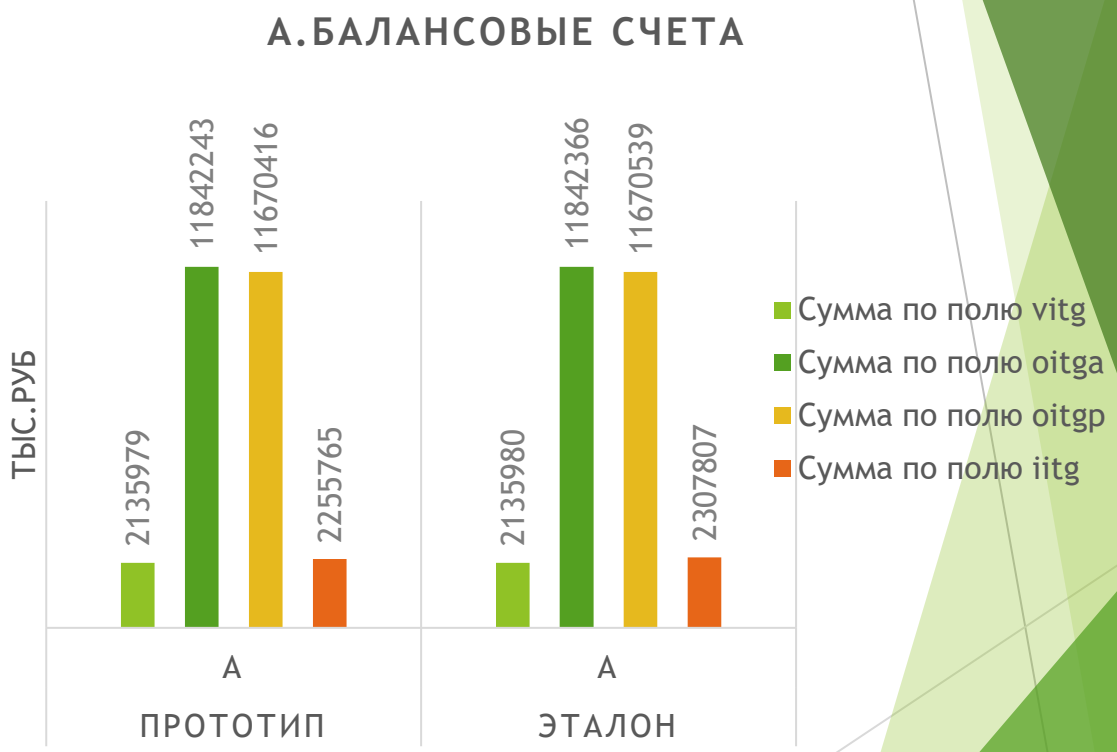
Раздел А. Балансовые счета:

А. Балансовые счета	Сумма по полю vitg	Сумма по полю oitga	Сумма по полю oitgp	Сумма по полю iitg
Прототип	**4271961**	**15367684**	**15367684**	**4563571**
а	2135979	11842243	11670416	2255765
р	2135982	3525441	3697268	2307806
Эталон	**4271960**	**15367807**	**15367807**	**4615614**
а	2135980	11842366	11670539	2307807
р	2135980	3525441	3697268	2307807

А. Балансовые счета	Сумма по полю vitg	Сумма по полю oitga	Сумма по полю oitgp	Сумма по полю iitg
Прототип	4271961	15367684	15367684	4563571
Эталон	4271960	15367807	15367807	4615614
Отклонение	1	-123	-123	-52043

Анализ в разрезе счетов:

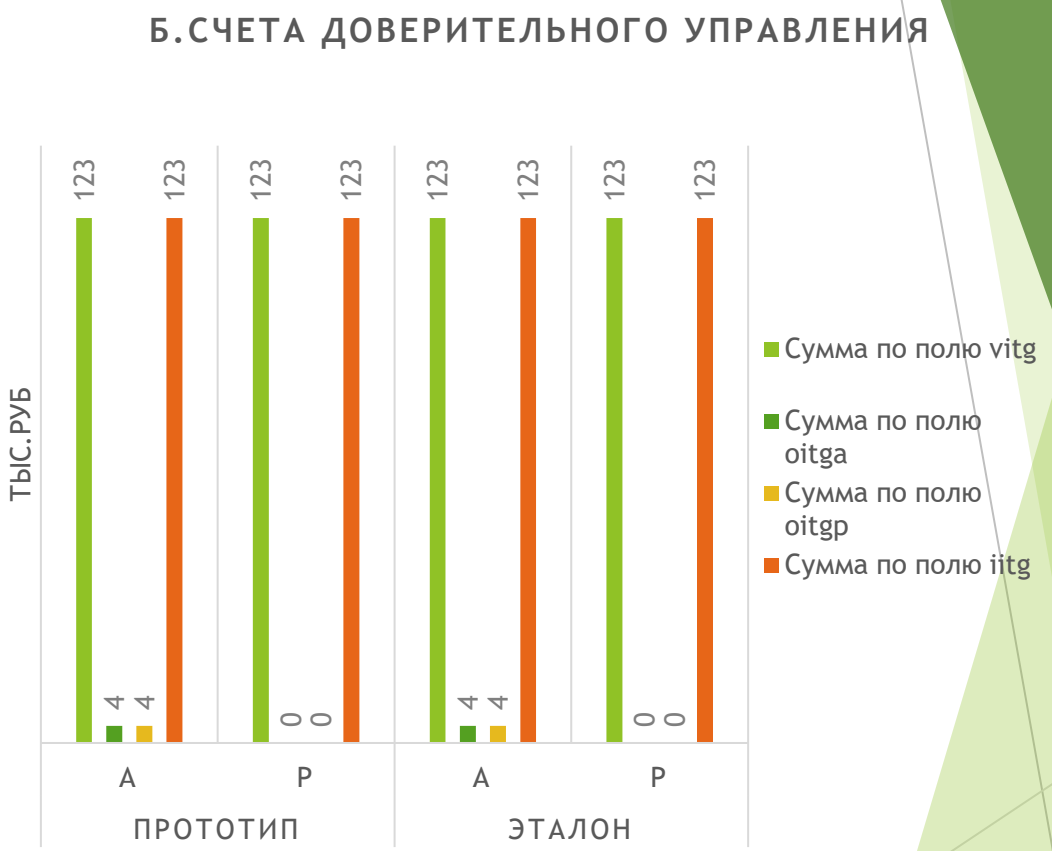
Активные счета	Прототип	Эталон	Отклонение
20202	68735	120778	-52043
61209		0	0
61212		0	0



Раздел Б. Счета доверительного управления: отклонений не выявлено.

Б.Счета доверительного управления	Сумма по полю vitg	Сумма по полю oitga	Сумма по полю oitgp	Сумма по полю iitg
Прототип	**246**	**4**	**4**	**246**
а	123	4	4	123
р	123	0	0	123
Эталон	**246**	**4**	**4**	**246**
а	123	4	4	123
р	123	0	0	123

Б. Счета доверительного управления	Сумма по полю vitg	Сумма по полю oitga	Сумма по полю oitgp	Сумма по полю iitg
Прототип	246	4	4	246
Эталон	246	4	4	246
Отклонение	0	0	0	0

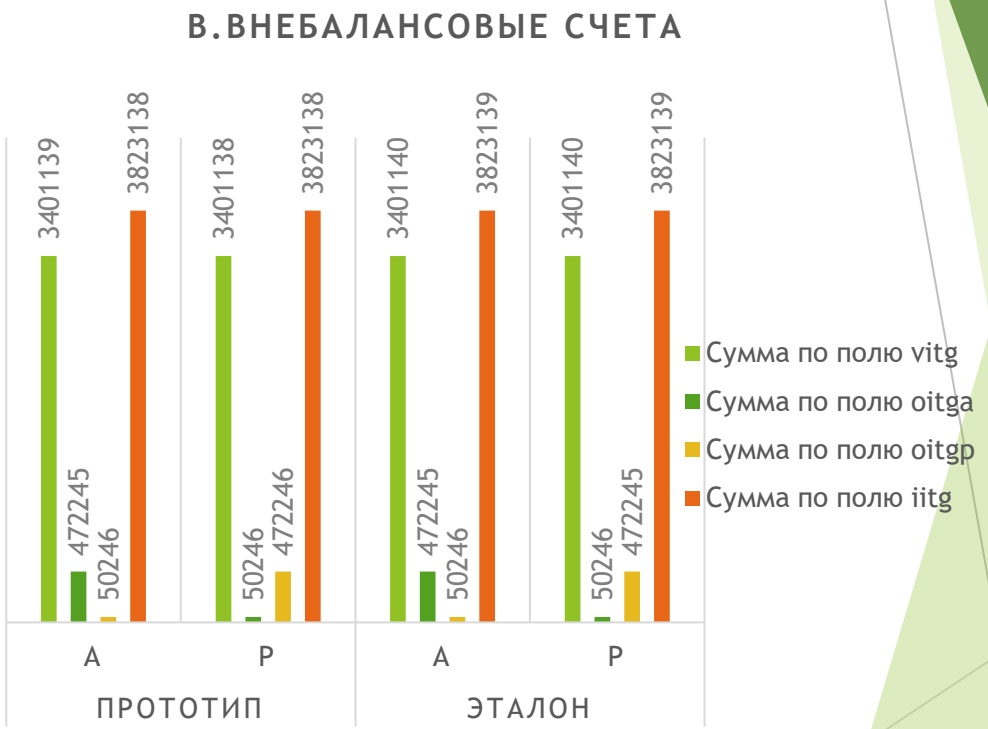


Раздел В. Внебалансовые счета: отклонение в следствии округления

В.Внебалансовые счета	Сумма по полю vitg	Сумма по полю oitga	Сумма по полю oitgp	Сумма по полю iitg
Прототип	**6802277**	**522491**	**522492**	**7646276**
а	3401139	472245	50246	3823138
р	3401138	50246	472246	3823138
Эталон	**6802280**	**522491**	**522491**	**7646278**
а	3401140	472245	50246	3823139
р	3401140	50246	472245	3823139

ТЫС.РУБ

В.Внебалансовые счета	Сумма по полю vitg	Сумма по полю oitga	Сумма по полю oitgp	Сумма по полю iitg
Прототип	6802277	522491	522492	7646276
Эталон	6802280	522491	522491	7646278
Отклонение	-3	0	1	-2



Итог

- Выявлено, что по счету 20202 ошибочно дублируются счета покрытия в таблице rest, поэтому неправильно отражается исходящие остатки в иностранной валюте и исходящий остаток итого (IC и IITG)

  В соответствии со статьей 28 ФЗ №173-ФЗ «О валютном регулировании» и Положении Банка России от 24.11.2022 N 809-П (ред. от 02.11.2024) «О Плане счетов бухгалтерского учета для кредитных организаций и порядке его применения» счет 20202 отражает наличность банка, в том числе и валюту;

  Дополнительно по счету 20202, счет покрытия не дублируется на валютные операции, а безналично резервируется на спецсчете покрытия.

  Исправление: счета покрытия имеют Code_Currency = 0 и имеют ошибочные суммы, а счета в иностранной валюте имеют Code_Currency = 978 и 840 и верные суммы операций, поэтому счета с Code_Currency = 0 полностью удаляем, а у валютных счетов меняем Code_Currency на 0.

- А так же выявлено что в прототипе, построенным мной, не выводятся счета 61209 и 61212, из за ошибки в БД, непосредственно в таблице bank.account_dbt. По этим счетам не заполнен kind_account. Поэтому нам достаточно внести информацию о типе счетов в таблицу.

  Запрос для исправления:

```
UPDATE bank.account_dbt
SET Kind_Account = 'A'
WHERE (account LIKE '61209%' OR account LIKE '61212%')
  AND (
      Kind_Account = '-'
      OR Kind_Account = ''
      OR Kind_Account IS NULL
```

Общие впечатления

Этап формирования ТЗ	На этапе формирования ТЗ я сталкивалась с большим количеством вопросов. Особенно сложным для понимания показался этап маппинга. Как новичку, мне потребовалось время, чтобы разобраться, как происходит процесс сопоставления и преобразования данных от исходных к целевым. Но с момента, как я разобралась с этим процессом, многое стало на свои места, и дальнейшая работа меня захватила. Особенно мне понравилась работа с базой данных, её анализ, выявление с помощью запросов закономерностей.
Этап прототипирования	На этом этапе я полностью погрузилась в построение поэтапного запроса, который будет соответствовать всем требованиям. Построив фактически весь запрос, я столкнулась по-настоящему со сложной задачей, так как мой прототип не выводил соответствующее эталону количество счетов из-за того, что я не учла один из критериев отбора (активные счета в заданном периоде). На эту задачу у меня ушло больше всего времени, но в конечном счёте я пришла к верной фильтрации и начала этап сравнения эталона с прототипом с целью выявления ошибок в БД. Это стало для меня самой простой задачей. Мне понадобилось обратиться к ФЗ и к базе.
Проверка прототипа и визуализация данных	Вся информация у меня была на руках, осталось грамотно её построить в Power BI. Так как с этим инструментом я начала работать впервые, я встретилась с трудностями, которые требуют углубления в инструмент и в специфику языка DAX. Попробовав построить графики для анализа, грамотного визуала у меня не выходило из-за нехватки знаний и времени. Поэтому я выбрала инструмент Excel. Но обучение инструменту BI я выделяю приоритетное место, ведь он даёт намного больше возможностей, чем Excel.

Проект для меня был по-настоящему интересным. Я получила удовольствие от его выполнения и очень большой пласт знаний и практических закреплений, повысила свой уровень SQL.

Спасибо за базу для его выполнения и кураторство!

Спасибо за внимание

Приложение 1

Список счетов, актуальных в заданном периоде дат:

```
SELECT distinct '2021-05-30' as date_from
  ,'2021-05-31' as date_to
  , account as acc_num
  , balance as acc_bal_num
  , case
      when kind_Account = 'А' then 'a'
      when kind_Account = 'П' then 'p'
      end as active_passiv_flag
FROM bank.account_dbt
where Open_Date<= '2021-05-31' and Close_Date>='2021-05-30'
```

Дата последнего движения по счету, по состоянию на определенную дату:

```
select distinct '2021-05-01' as date
      , ac.account  as acc_num
      ,(select max(ar.date_carry)
            from bank.arhdoc_dbt as ar
            join bank.account_dbt as ac on ar.Chapter = ac.Chapter
             and ar.Real_Payer = ac.account
             where ac.account = '20202810000090000001' and ar.date_carry between ac.open_date
             and '2021-05-01') as Date_last_active
from bank.arhdoc_dbt as ar
  join bank.account_dbt as ac on ar.Chapter = ac.Chapter
  and ar.Real_Payer = ac.account
where ac.account = '20202810000090000001'
```

Исходящий остаток по счету на определенную дату, в рублях:

```
select '2021-05-30' as date
        , account  as acc_num
        , rest as rest_out
from bank.restdate_dbt
where  code_currency = 0 and account = '20202810000090000001'
and case
        when date_carry<>'2021-05-30' then date_carry = (select max(date_carry) from bank.restdate_dbt
        where date_carry < '2021-05-30')
        else date_carry = (select max(date_carry) from bank.restdate_dbt where date_carry <= '2021-05-30')
    end
```

Кредитовый оборот по счету в заданном периоде дат, в рублях:

```
Select   '2021-05-01' as date_from
        ,'2021-05-31' as date_to
        , ac.account as acc_num,
        case
           when ac.Kind_Account = 'А' then (
               select sum(ar.sum)
               from bank.account_dbt as ac
               join bank.arhdoc_dbt as ar  on ac.account = ar.Real_Receiver
               where ac.code_currency = 0 and ac.account = '20202810000320000001' and ar.date_carry between '2021-05-01' and '2021-05-31')
           when ac.Kind_Account = 'П' then (
               select sum(ar.sum)
               from bank.account_dbt as ac
               join bank.arhdoc_dbt as ar  on ac.account = ar.Real_Payer
               where ac.code_currency = 0 and ac.account = '20202810000320000001' and ar.date_carry between '2021-05-01' and '2021-05-31')
           end as ct_turnover,
           ac.k5
from bank.account_dbt as ac
join bank.arhdoc_dbt as ar  on ac.account = ar.Real_Receiver
where ac.account = '20202810000320000001'
group by  ac.account, ac.k5, ac.Kind_Account
```

Дебетового оборота по счету, в заданном периоде дат, в рублях:

```
Select   '2021-05-01' as date_from
        ,'2021-05-31' as date_to
        , ac.account as acc_num,
        case
           when ac.Kind_Account = 'А' then (
               select sum(ar.sum)
               from bank.account_dbt as ac
               join bank.arhdoc_dbt as ar  on ac.account = ar.Real_Payer
               where ac.code_currency = 0 and ac.account = '20202810000320000001' and ar.date_carry between '2021-05-01' and '2021-05-31')
           when ac.Kind_Account = 'П' then (
               select sum(ar.sum)
               from bank.account_dbt as ac
               join bank.arhdoc_dbt as ar  on ac.account = ar.Real_Receiver
               where ac.code_currency = 0 and ac.account = '20202810000320000001' and ar.date_carry between '2021-05-01' and '2021-05-31')
           end as dt_turnover,
           ac.d5
from bank.account_dbt as ac
join bank.arhdoc_dbt as ar  on ac.account = ar.Real_Receiver
where ac.account = '20202810000320000001'
group by  ac.account, ac.d5, ac.Kind_Account
```