数据连接器集成手册

Yeastar P 系列云 PBX

版本: 1.0
日期: 2026年05月25日

400-661-8815
0592-5503307
www.yeastar.cn

Contents

数据连接器集成概述

Yeastar P 系列云 PBX 支持通过数据连接器将 PBX 数据导出至第三方 SQL 数据库 (例如 PostgreSQL、MySQL 或 Microsoft SQL)。同步至数据库中的数据可通过商业智能 (Business Intelligence, BI) 工具、计费系统或其他应用读取，用于统计分析和报表展示。

使用要求

服务器	要求
Yeastar PBX	84.23.0.83 或更高版本。
第三方数据库	使用以下数据库： • PostgreSQL • MySQL • Microsoft SQL

支持同步的数据范围

数据	说明
通话记录	同步通话记录列表、通话阶段及时间线数据。
录音记录	同步录音记录。 **Note:** 不包含录音文件。
系统指标	同步仪表盘中的以下系统指标数据。 • **当前通话数** • **CPU 使用率** • **内存使用率** • **本地存储使用率** • **已注册分机** • **可用 SIP 中继** • **Linkus 客户端登录** **Note:** 系统会在同步时实时采集系统指标数据。
通话备注	同步话务标签和备注内容。

数据	说明
报告记录	同步通话报告数据。
AI 总结 & AI 转录	同步 AI 生成的总结与转录内容。

操作指南

Yeastar P 系列云 PBX 与数据连接器的集成较为简单，仅需配置数据库连接信息和同步策略，即可将 PBX 数据同步至第三方数据库。

更多信息，请参见 通过数据连接器连接 Yeastar P 系列云 PBX 和 SQL 数据库。

Note:
完成集成后，可将第三方数据库接入商业智能 (Business Intelligence, BI) 工具、计费系统或其他应用，实现数据可视化和分析。我们提供了第三方数据库与 Grafana、Power BI 集成的详细说明。更多信息，请参见 将数据库中的 PBX 数据导入 Grafana 和 将数据库中的 PBX 数据导入 Power BI。

通过数据连接器连接 Yeastar P 系列云 PBX 和 SQL 数据库

本文介绍如何在 Yeastar P 系列云 PBX 上配置数据连接器，将 PBX 数据导出至第三方 SQL 数据库。

使用要求

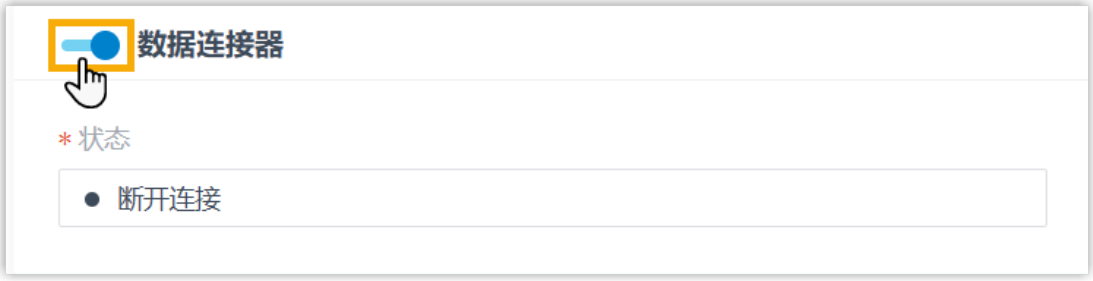
Yeastar P 系列云 PBX 的固件为 84.23.0.83 或更高版本。

前提条件

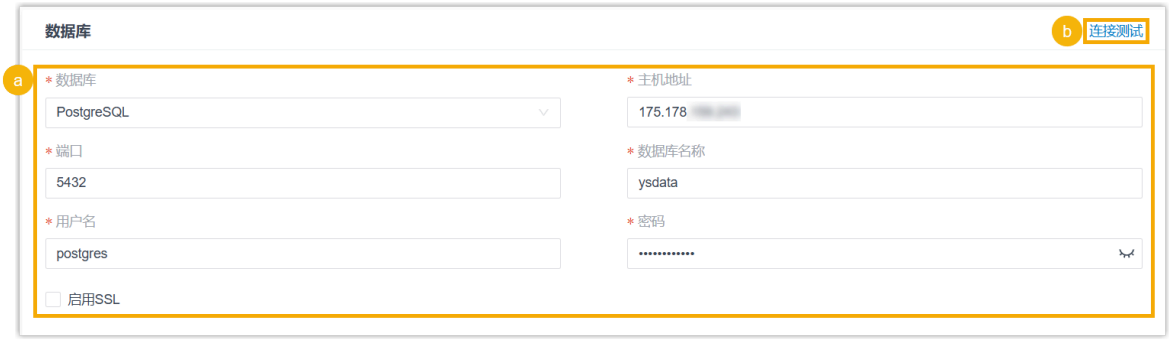
- 已部署以下任一数据库系统：
  - PostgreSQL
  - MySQL
  - Microsoft SQL
- 已创建用于存储 PBX 数据的数据库。

操作步骤

1. 登录 PBX 管理网页，进入 **应用对接 > 数据连接器**。
2. 打开 **数据连接器** 开关。



3. 在 **数据库** 栏，配置第三方数据库连接信息。



a. 完成以下设置，以连接数据库。

配置项	说明
数据库	选择数据库类型。 • **PostgreSQL** • **MySQL** • **Microsoft SQL**
主机地址	填写数据库服务器的 IP 地址或域名。
端口	填写数据库端口。
数据库名称	填写数据库名称。
用户名	填写用于连接数据库的用户名。
密码	填写用于连接数据库的密码。
启用 SSL	启用或禁用 SSL 证书校验。 若启用，点击 **浏览** 选择并上传 SSL 证书。

b. **可选：** 点击 **连接测试**，验证 PBX 是否可以成功连接数据库。

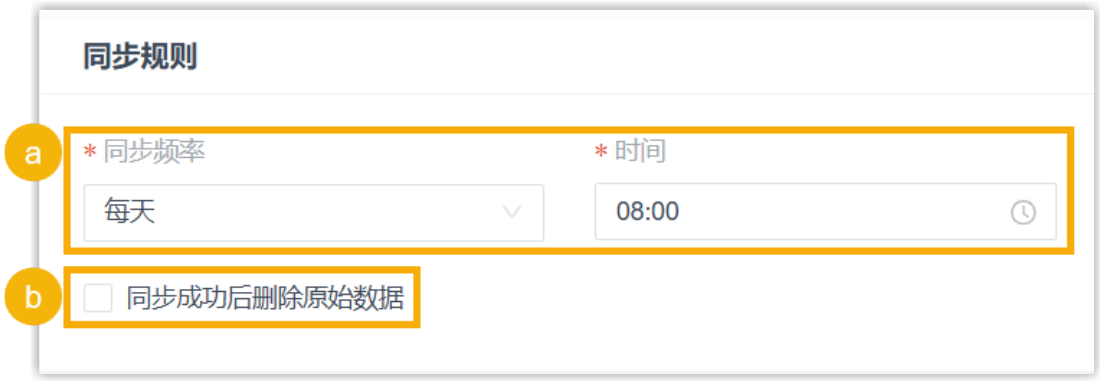
如果显示 “连接测试成功”，表示数据库连接成功。

4. 在 **数据范围** 栏，选择需要同步到第三方数据库的数据类型。

数据	说明
通话记录	同步通话记录列表、通话阶段及时间线数据。
录音记录	同步录音记录。 **Note:** 不包含录音文件。
系统指标	同步仪表盘中的以下系统指标数据。 • **当前通话数** • **CPU 使用率** • **内存使用率**

数据	说明
	• **本地存储使用率** • **已注册分机** • **可用 SIP 中继** • **Linkus 客户端登录** **Note:** 系统会在同步时实时采集系统指标数据。
通话备注	同步话务标签和备注内容。
报告记录	同步通话报告数据。
AI 总结 & AI 转录	同步 AI 生成的总结与转录内容。

5. 在 **同步规则** 栏，设置数据同步频率及自动清理规则。



   a. 在 **同步频率** 下拉列表中，选择数据同步到第三方数据库的频率。

   Note:
   系统将基于默认时区，根据所选同步周期进行数据同步。

   b. **可选：** 如需在同步完成后删除系统中的源数据，勾选 **同步成功后删除原始数据**。

   Note:
   通话记录源数据一旦删除，将无法在 PBX 上访问对应的通话记录。

6. 执行首次数据同步操作，将数据同步至第三方数据库。
   a. 点击 **立即同步**。
   b. 在弹出的窗口中，点击 **确定** 并确认操作。

   等待数据同步完成。

执行结果

对接状态显示为 **已连接**，表示 PBX 已成功连接数据库，可向数据库单向同步数据。

Note:
可将第三方数据库接入商业智能 (Business Intelligence, BI) 工具、计费系统或其他应用，实现数据可视化和分析。我们提供了第三方数据库与 Grafana 和 Power BI 集成的详细说明。更多信息，请参见 将数据库中的 PBX 数据导入 Grafana 和 将数据库中的 PBX 数据导入 Power BI。

将数据库中的 PBX 数据导入商业智能(BI)工具

将数据库中的 PBX 数据导入 Grafana

PBX 数据同步至第三方数据库后，可将该数据库添加为 Grafana 数据源，并使用 Yeastar 提供的仪表板模板在 Grafana 上导入，实现数据可视化。

使用要求

平台	要求
Grafana	具有 **Organization administrator** 角色。
第三方数据库	• **版本**：使用 Grafana 支持的数据库版本。 更多信息，请参见以下官方链接： ◦ 支持的 Microsoft SQL Server 版本 ◦ 支持的 MySQL 数据库 ◦ 支持的 PostgreSQL 数据库 • **网络**：数据库与 Grafana 网络互通。 • **帐户**：使用具有只读权限 (SELECT) 的帐户。 **Note:** Grafana 不会校验查询的安全性，因此用户可能执行存在风险的 SQL 语句。建议创建一个权限受限的专用账户，以降低安全风险。

前提条件

已将 PBX 数据同步至第三方数据库。

操作步骤

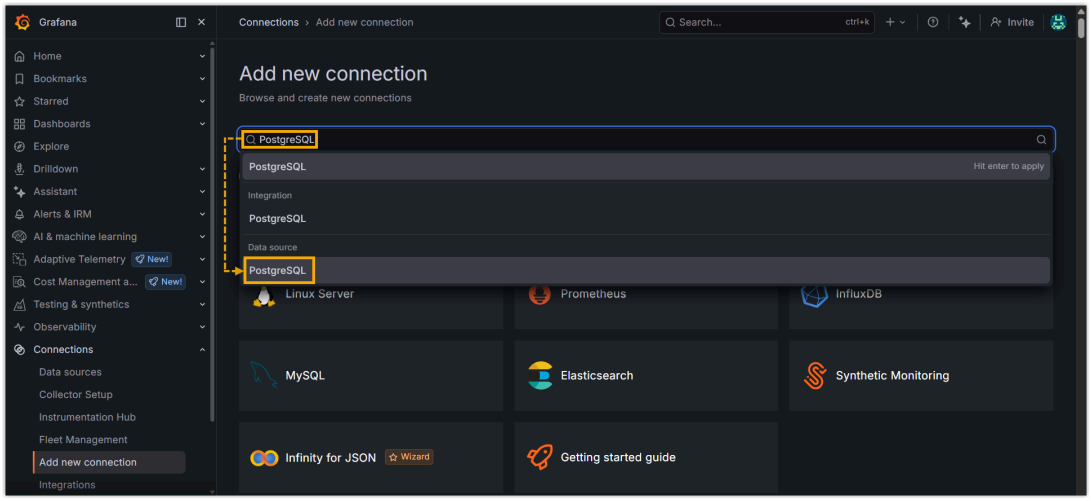
Note:
本文以 **PostgreSQL** 为例，介绍如何将数据库中的 PBX 数据导入 Grafana。Microsoft SQL Server 和 MySQL 的操作基本相同。

- 步骤一、连接 Grafana 与数据库

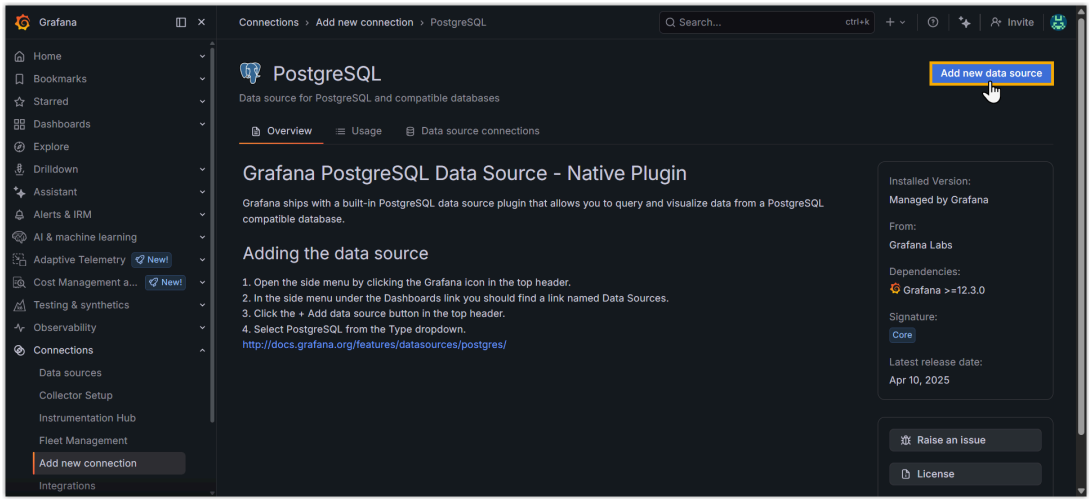
步骤一、连接 Grafana 与数据库

1. 登录 Grafana 平台，进入 **Connections > Add new connection**。
2. 添加数据源。
   a. 搜索数据库类型，并从数据源列表中选择数据库类型。

   在本例中，选择 **PostgreSQL** 数据源。



   b. 点击右上角的 **Add new data source**。



3. 填写以下信息，以连接数据库。
   a. 在 **Connection** 栏，填写数据库连接信息。

配置项	说明
Host URL	填写数据库服务器的 IP 地址/域名及端口。

配置项	说明
Database name	填写数据库名称。

b. 在 **Authentication** 栏，填写认证信息。

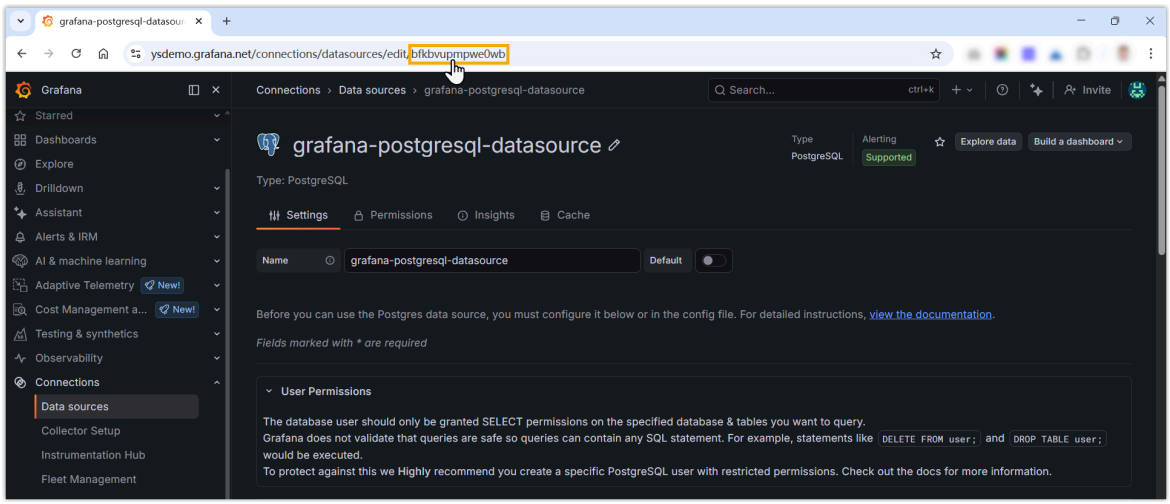
配置项	说明
Username	填写用于连接数据库的用户名。
Password	填写用于连接数据库的密码。

4. **可选：** 按需完成其他设置。
5. 滚动到页面底部，点击 **Save & test** 测试并保存数据源连接。

   如果显示 “Database Connection OK”，表示数据库连接成功。

步骤二、导入模板仪表盘

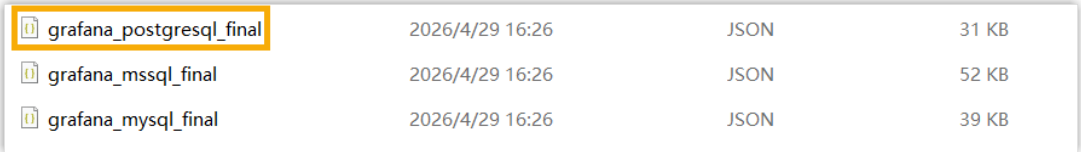
1. 在浏览器的地址栏，复制 UID (URL 的最后一段)。



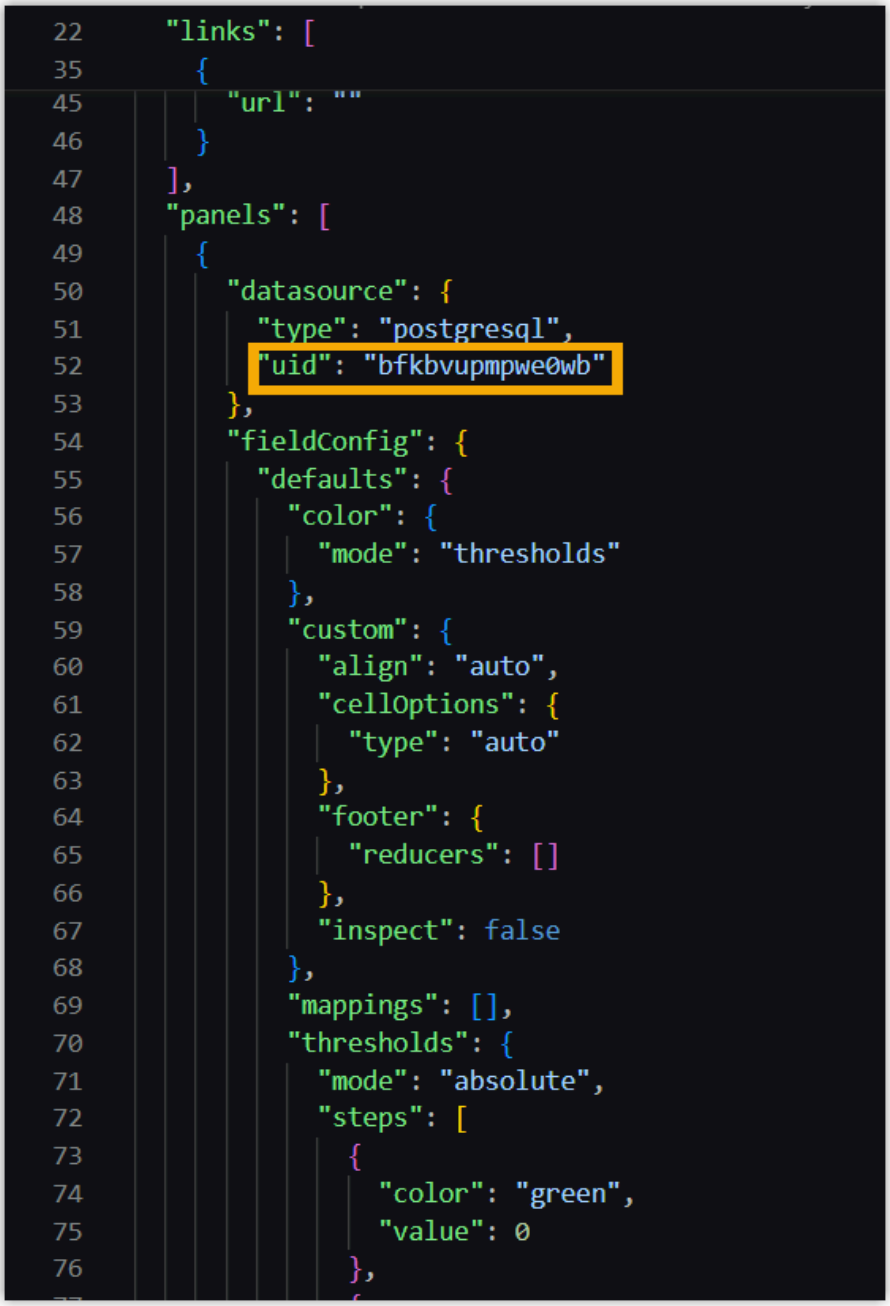
   在本例中，获取的值为 `bfkbvupmpwe0wb`。

2. 下载并更新仪表盘 JSON 模板。
   a. 下载并解压 仪表盘模板压缩包。该压缩包包含适用于不同数据库的模板，请根据数据库类型选择合适的模板。

      在本例中，我们将使用 PostgreSQL 的模板。

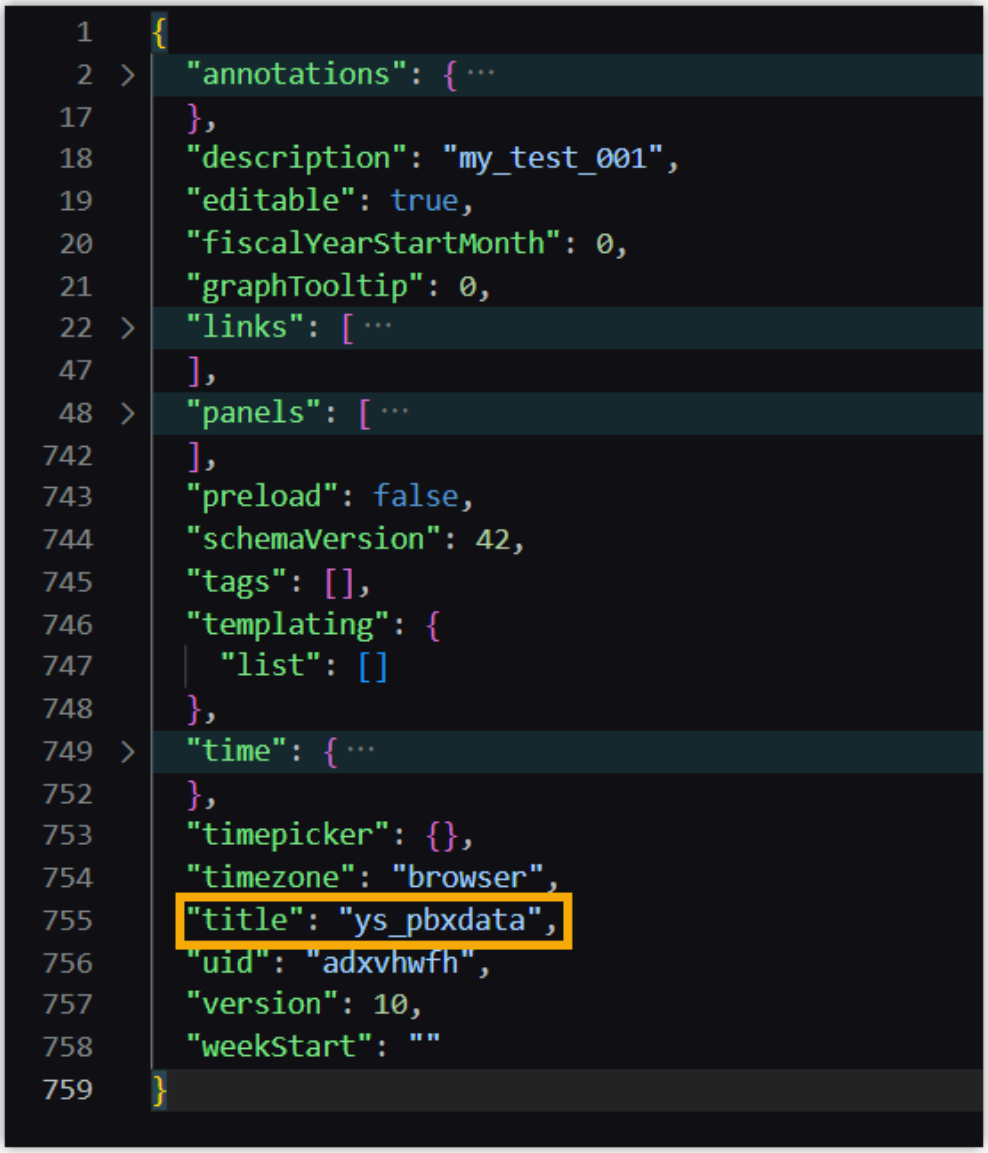


   b. 将每个面板中 `datasource` 下的所有 `uid` 值替换为获取到的值。



Note:
保持 annotations 和 dashboard 中的 `uid` 值不变。

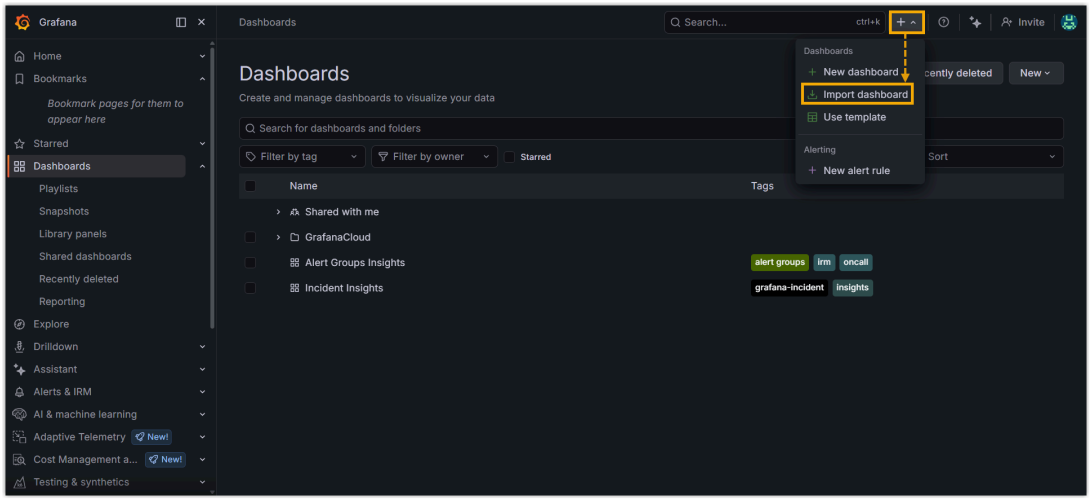
c. **可选：** 按需修改 `title` 值，该值将用作 Grafana 中的仪表板名称。



3. 将仪表盘模板导入到 Grafana。

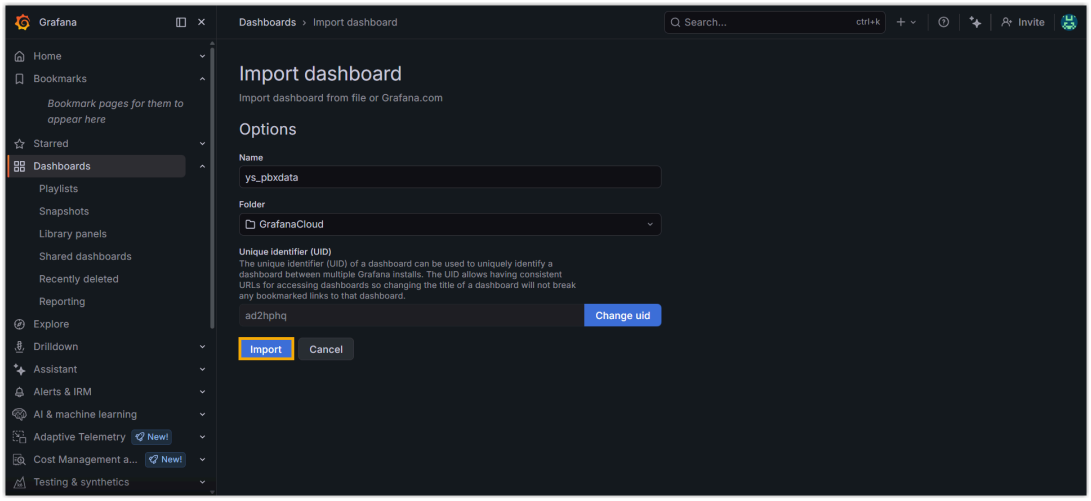
   a. 在左侧导航栏，进入 **Dashboards**。

   b. 在页面右上角，点击 ＋，选择 **Import dashboard**。



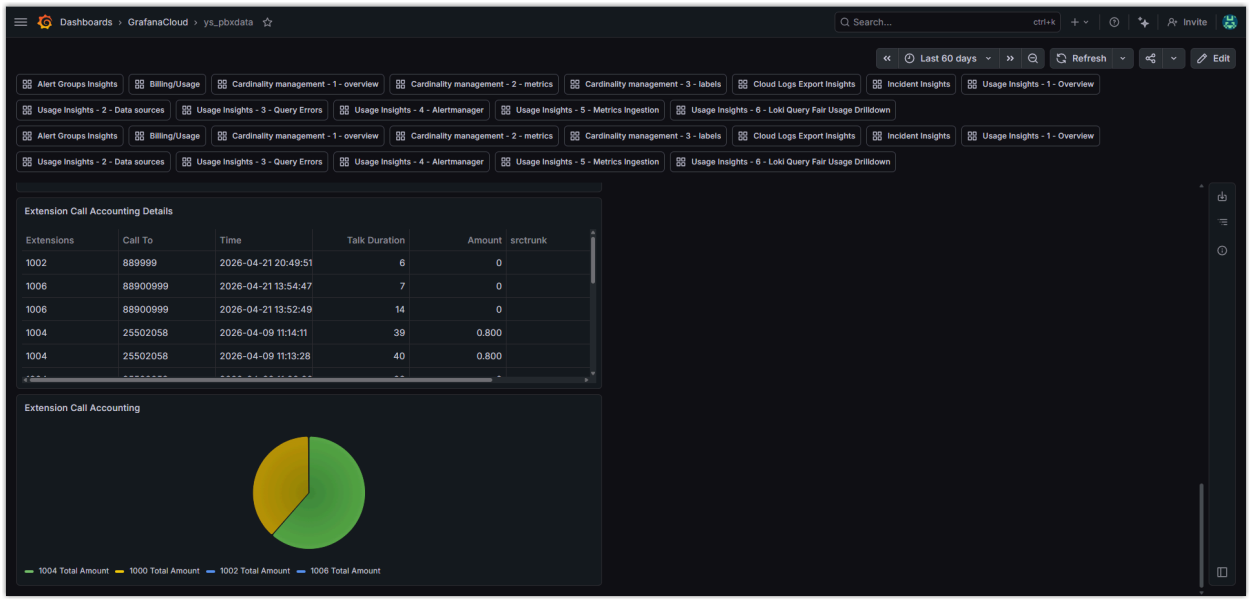
c. 点击 **Upload dashboard JSON file**，上传 `.json` 文件。

d. 点击 **Import**。



执行结果

仪表盘已成功导入，可在仪表盘中查看和分析数据。



后续操作

按需添加或修改 SQL 查询语句，以自定义要展示的数据。

Note:

由于 Grafana 不支持在单个数据源中执行组合查询，因此 Yeastar 提供的仪表盘模板仅支持展示以下通话报告数据。

- **分机通话计费**
- **分机通话计费详情**
- **AI 接待员通话活动**
- **坐席未接通报告**
- **队列回拨报告**
- **满意度报告**
- **满意度调查详情**
- **IVR报告**
- **DID/去电号码活动**

如需展示其他报告，可参考 基于多条 SQL 查询的通话报告计算逻辑 了解各报告的数据逻辑，并按需添加 SQL 查询语句。

将数据库中的 PBX 数据导入 Power BI

PBX 数据同步至第三方数据库后，可将该数据库添加为 Power BI 数据源，并使用 Yeastar 提供的模板在 Power BI 上导入，实现数据可视化。

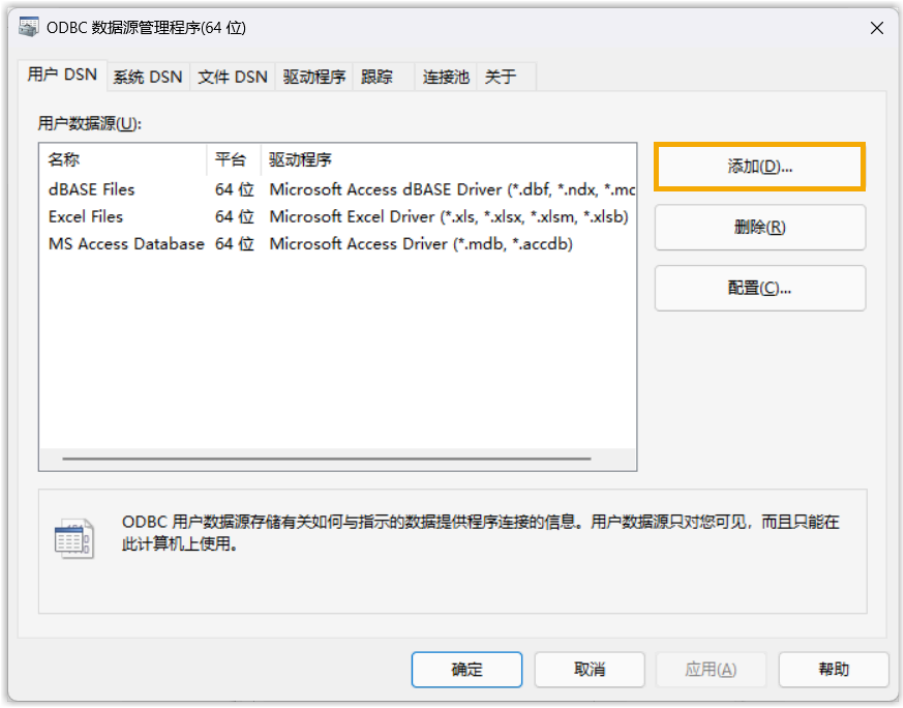
将 PBX 数据从 PostgreSQL 导入 Power BI

前提条件

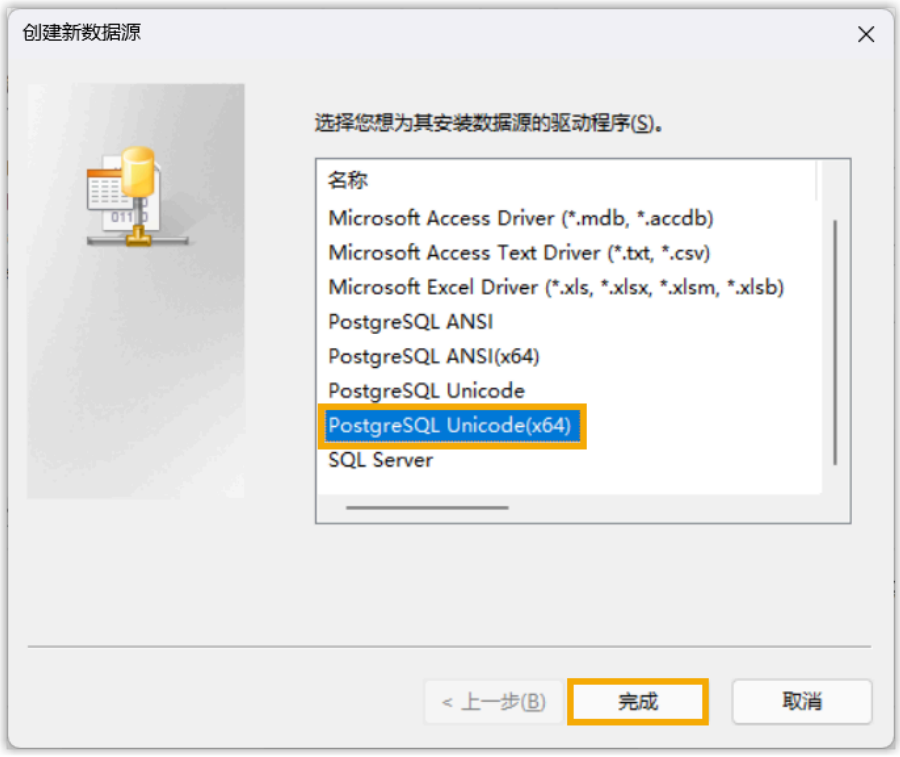
- 下载并解压 Power BI 模板。该压缩包包含适用于不同数据库的模板，请根据数据库类型选择合适的模板。
- 下载并安装 Power BI desktop 和 PostgreSQL ODBC 驱动程序。

操作步骤

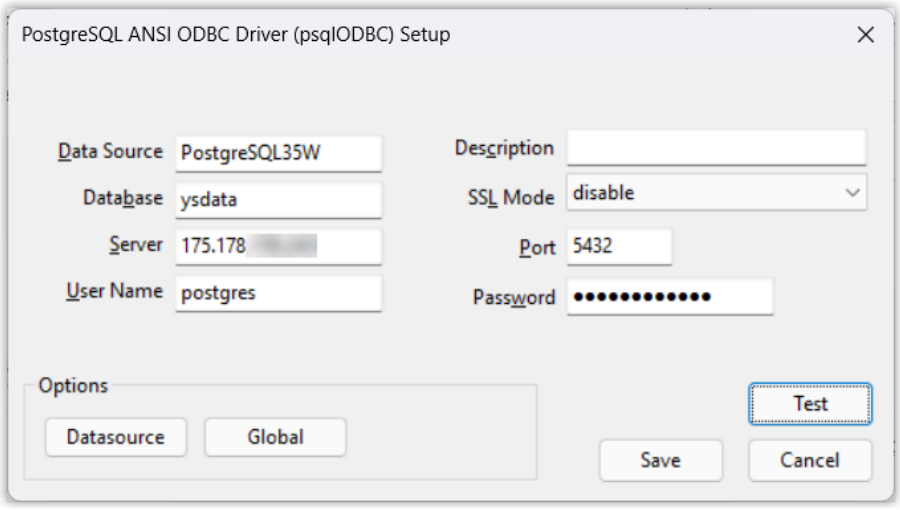
1. 配置数据库连接。
   a. 打开 **ODBC 数据源 (64位)** 管理器。
   b. 在 **用户 DSN** 页签下，点击 **添加**。



   c. 在弹出的窗口中，选择 **PostgreSQL Unicode(x64)**，点击 **完成**。



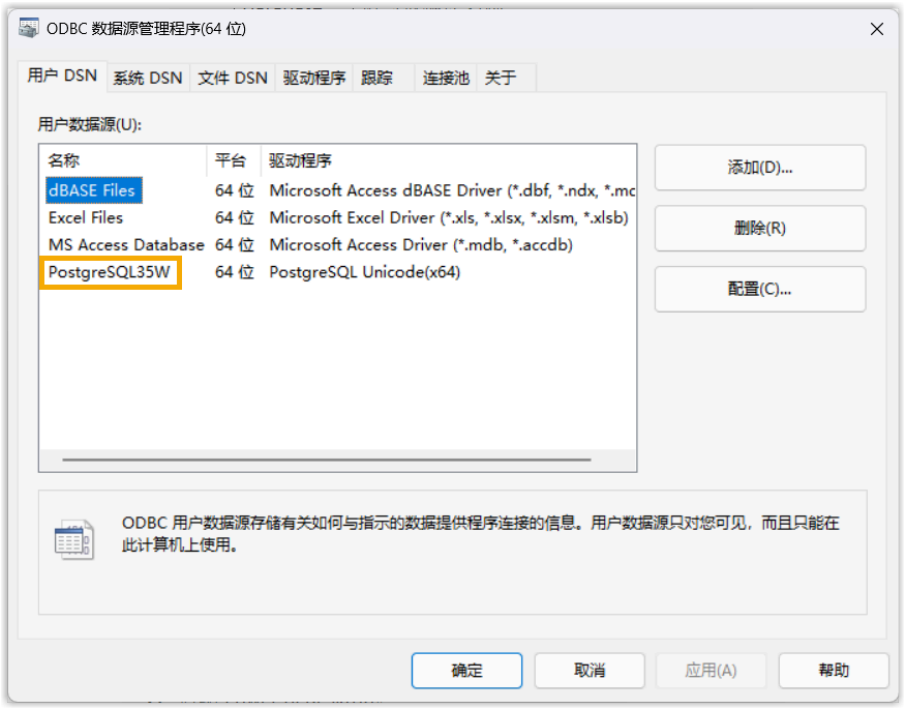
d. 填写以下信息，然后点击 **保存**。



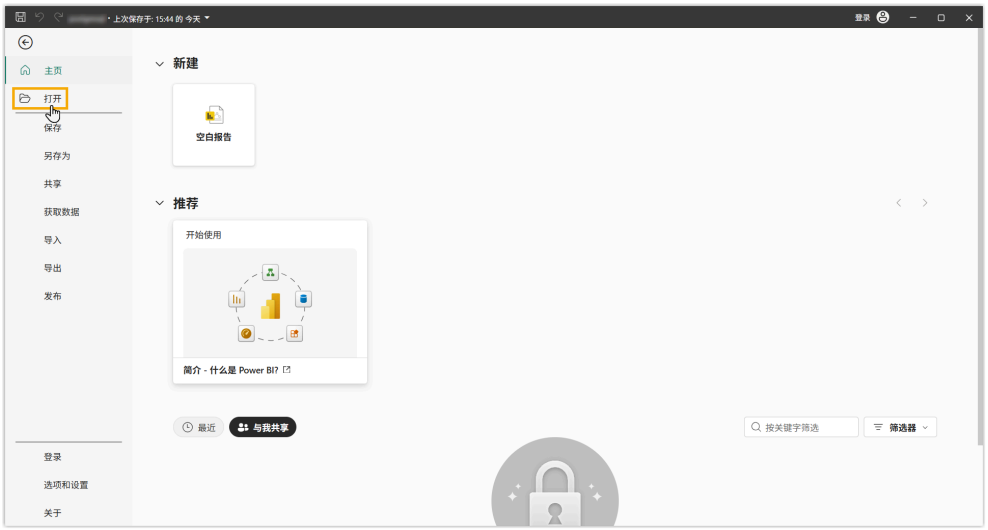
配置项	说明
Database	填写数据库名称。
Server	填写数据库服务器的 IP 地址或域名。
Port	填写数据库端口。

配置项	说明
User Name	填写用于连接数据库的用户名。
Password	填写用于连接数据库的密码。

e. 记住 DSN 名称，后续在 Power BI 中将使用该名称。

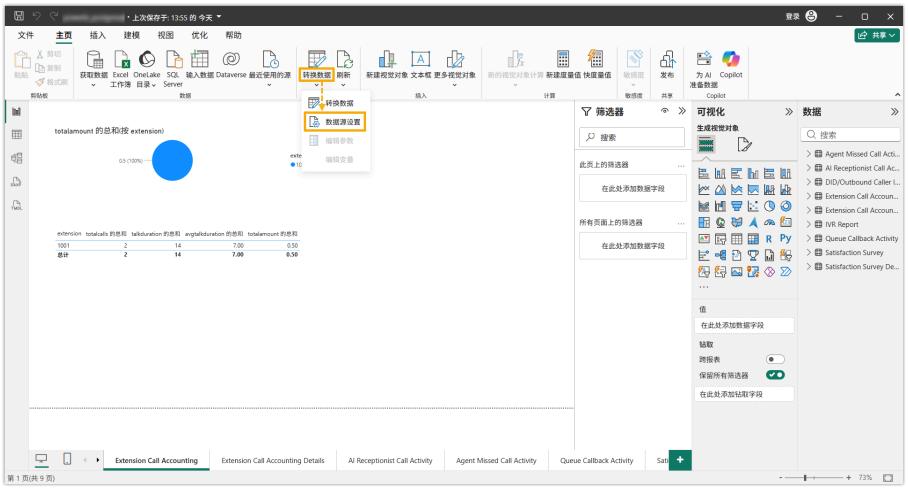


2. 启动 Power BI desktop。
3. 点击 **打开**，选择并打开 Yeastar 提供的模板文件。



4. 更改数据源并配置访问权限。

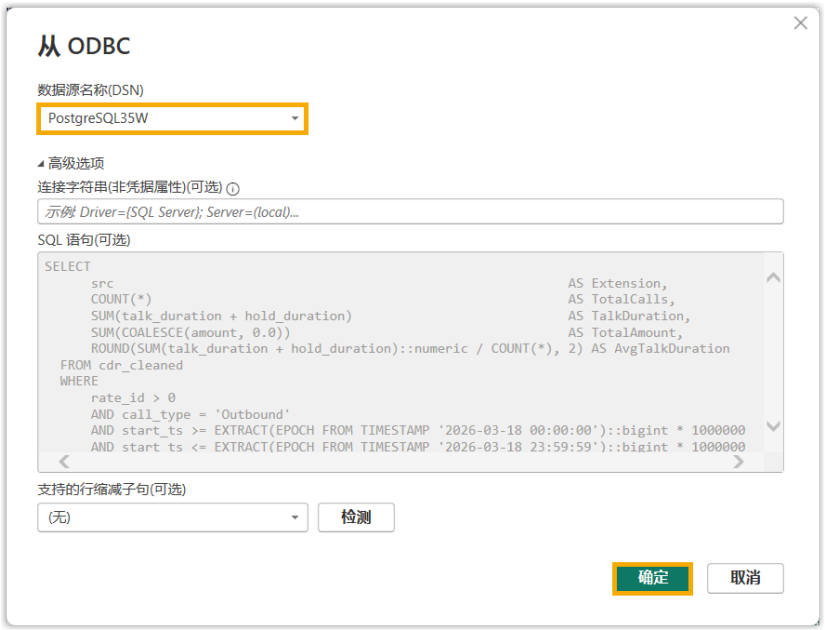
a. 在顶部工具栏，点击 **转换数据**，然后选择 **数据源设置**。



b. 更改数据源。

i. 选择默认数据源，点击 **更改源**。

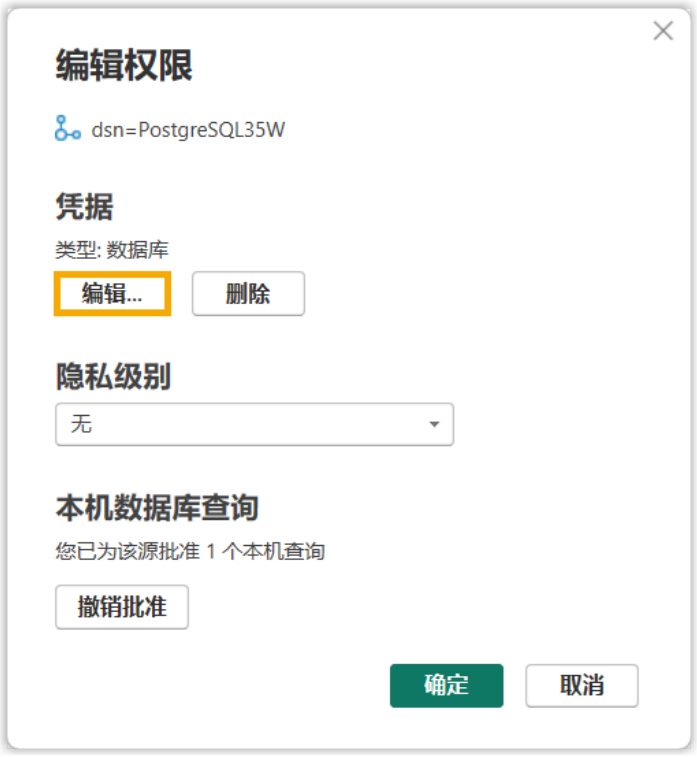
ii. 在 **数据源名称 (DSN)** 下拉列表中，选择对应数据库，然后点击 **确定**。



c. 配置数据库访问凭证。

i. 点击 **编辑权限**。

ii. 在弹出的窗口中，点击 **编辑**。

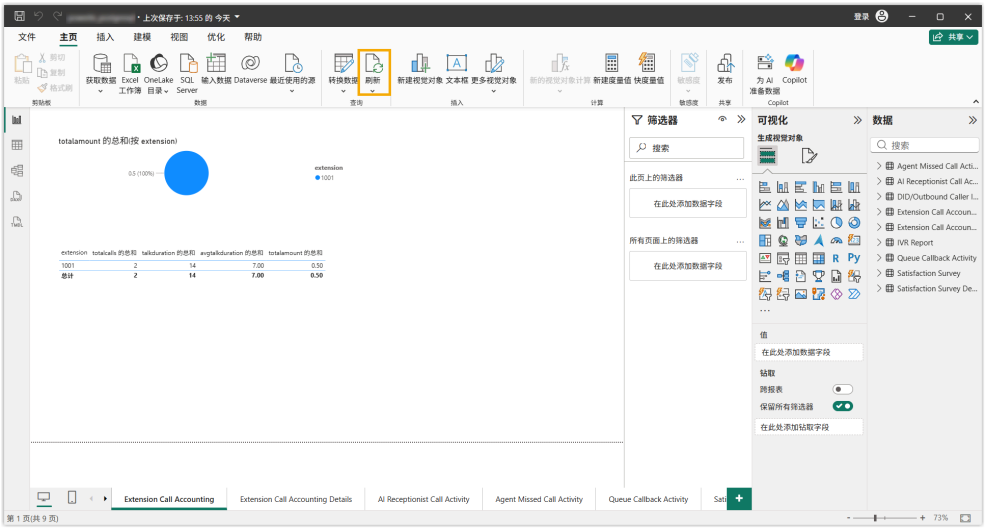


iii. 选择 **数据库**，填写认证信息，然后点击 **保存**。

配置项	说明
用户名	填写用于连接数据库的用户名。
密码	填写用于连接数据库的密码。

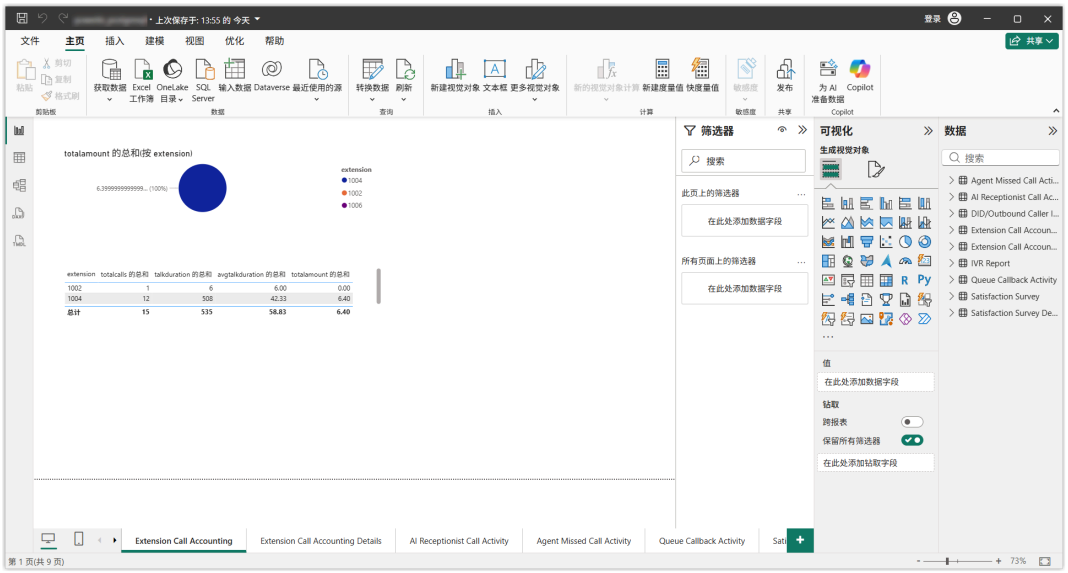
d. 保存设置。

5. 在页面顶部，点击 **刷新**，执行所有 SQL 查询。



执行结果

数据已成功导入 Power BI，实现数据可视化。



后续操作

按需添加或修改 SQL 查询语句，以自定义要展示的数据。

Note:
由于 Power BI 不支持在单个数据源中执行组合查询，因此 Yeastar 提供的仪表盘模板仅显示以下通话报告数据。

- **分机通话计费**

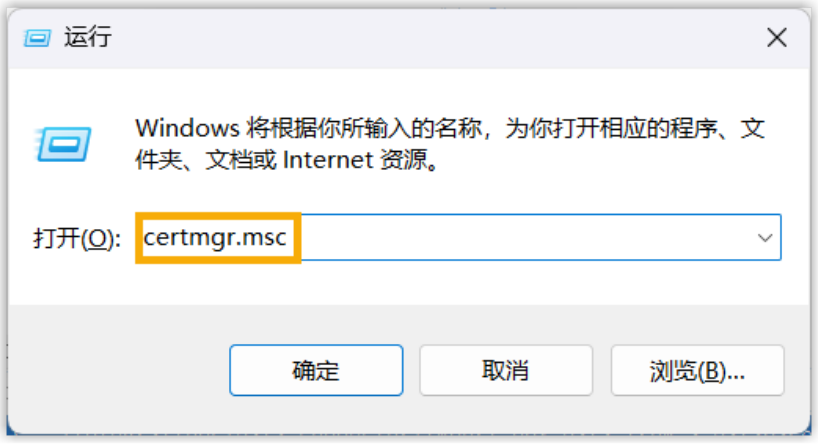
- **分机通话计费详情**
- **AI 接待员通话活动**
- **坐席未接通报告**
- **队列回拨报告**
- **满意度报告**
- **满意度调查详情**
- **IVR报告**
- **DID/去电号码活动**

如需展示其他报告，可参考 基于多条 SQL 查询的通话报告计算逻辑 了解各报告的数据逻辑，并按需添加 SQL 查询语句。

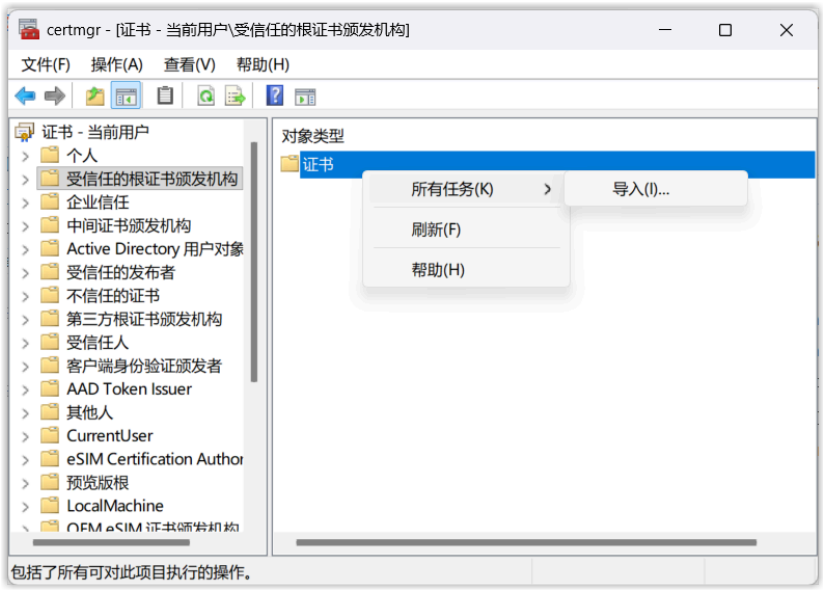
将 PBX 数据从 MySQL 导入 Power BI

前提条件

- 下载并解压 Power BI 模板。该压缩包包含适用于不同数据库的模板，请根据数据库类型选择合适的模板。
- 下载并安装 Power BI desktop 和 MySQL Connector/NET。
- (可选) 如果 MySQL 服务器启用了 SSL，需安装证书。
    1. 下载 SSL 公共证书。
    2. 在受信任的根证书颁发机构存储中安装 SSL 证书。
        a. 在电脑上运行 `certmgr.msc`，打开证书管理器。



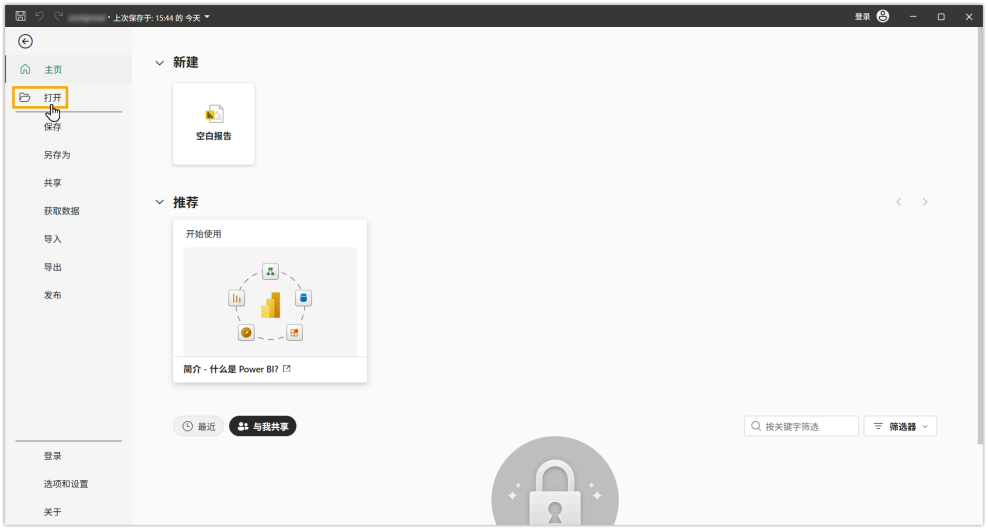
        b. 右键点击 **受信任的根证书颁发机构**，从 **所有任务** 列表中选择 **导入**。



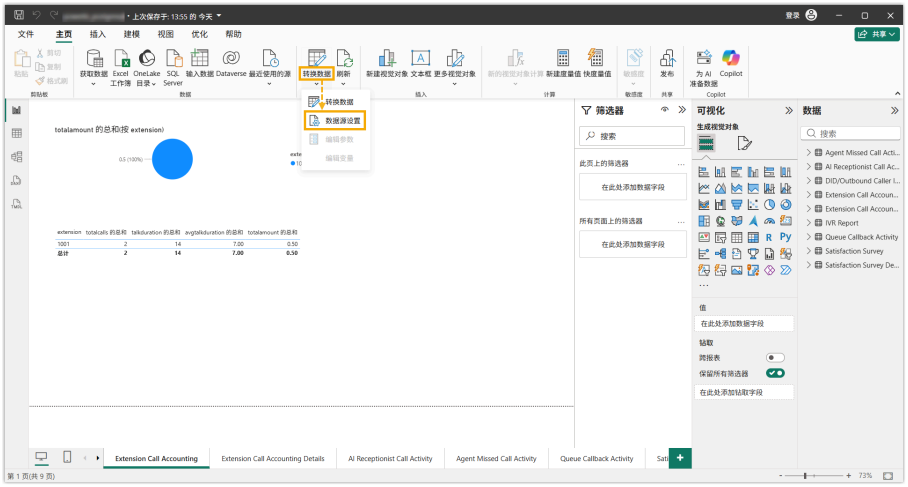
c. 按照向导中的提示导入根证书，然后选择 **确定**。

操作步骤

1. 启动 Power BI desktop。
2. 点击 **打开**，选择并打开 Yeastar 提供的模板文件。



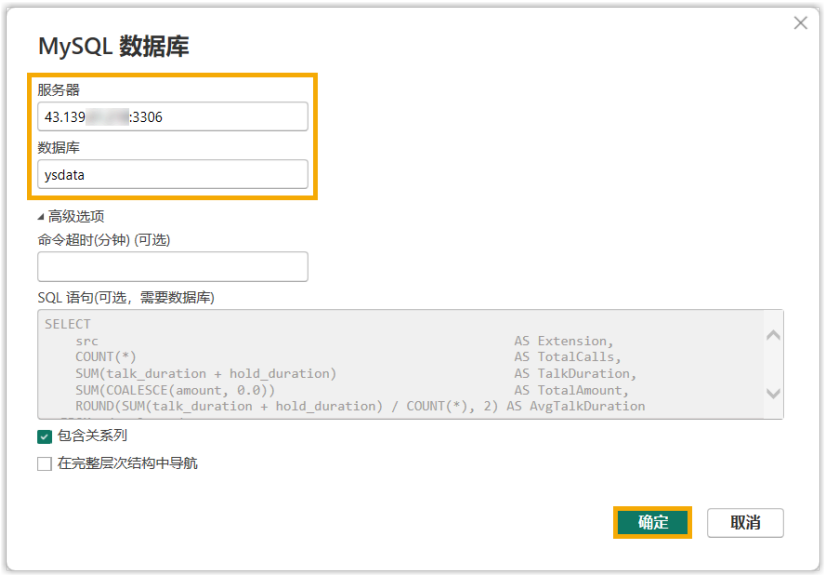
3. 更改数据源并配置访问权限。
    a. 在顶部工具栏，点击 **转换数据**，然后选择 **数据源设置**。



b. 更改数据源。

i. 选择默认的数据源，点击 **更改源**。

ii. 修改数据库连接信息，点击 **确定**。



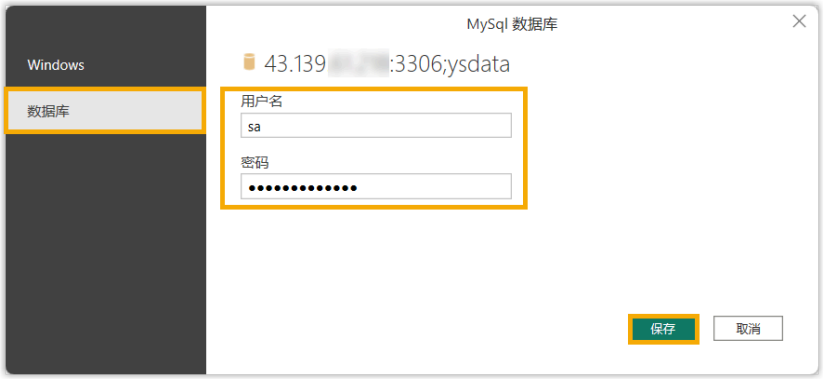
配置项	说明
服务器	填写数据库服务器的 IP 地址/域名及端口。
数据库	填写数据库名称。

c. 配置数据库访问凭证。

i. 点击 **编辑权限**。

ii. 在弹出的窗口中，点击 **编辑**。

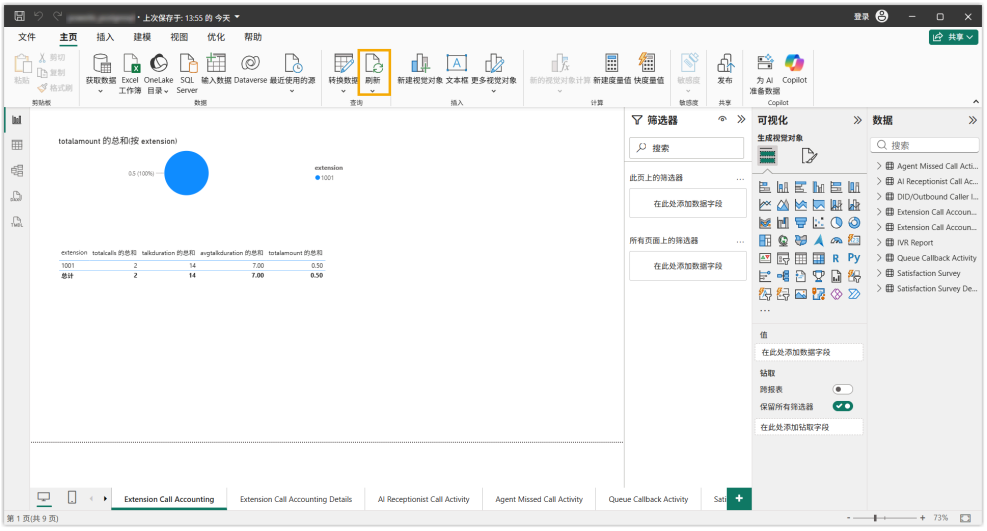
iii. 选择 **数据库**，填写认证信息，然后点击 **保存**。



配置项	说明
用户名	填写用于连接数据库的用户名。
密码	填写用于连接数据库的密码。

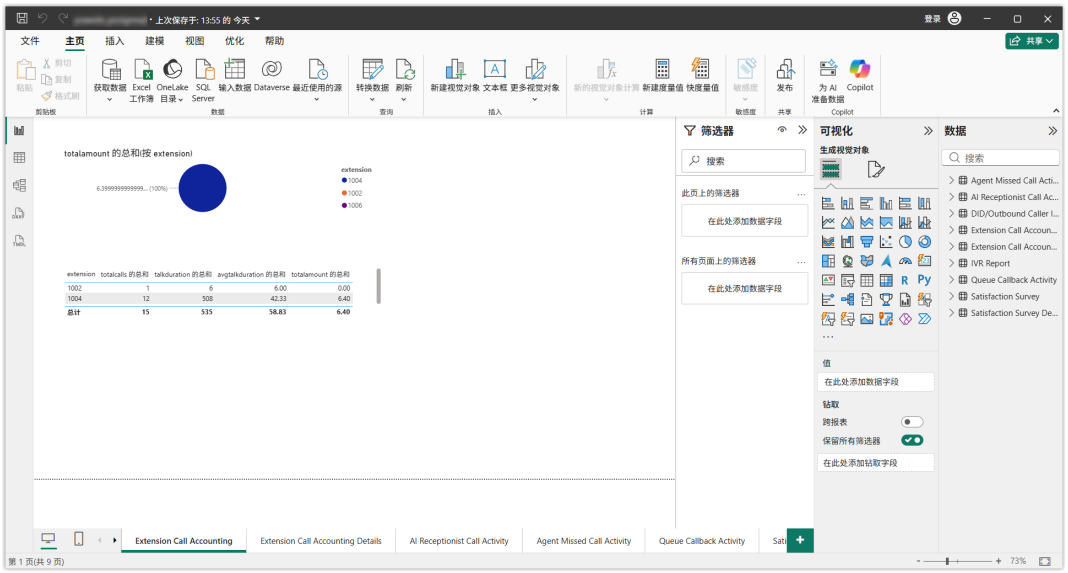
d. 保存设置。

4. 在页面顶部，点击 **刷新**，执行所有 SQL 查询。



执行结果

数据已成功导入 Power BI，实现数据可视化。



后续操作

按需添加或修改 SQL 查询语句，以自定义要展示的数据。

Note:

由于 Power BI 不支持在单个数据源中执行组合查询，因此 Yeastar 提供的仪表盘模板仅显示以下通话报告数据。

- **分机通话计费**

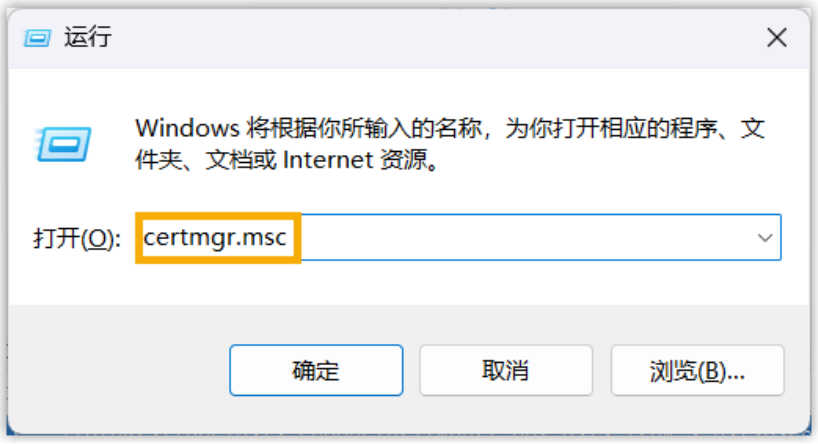
- **分机通话计费详情**
- **AI 接待员通话活动**
- **坐席未接通报告**
- **队列回拨报告**
- **满意度报告**
- **满意度调查详情**
- **IVR报告**
- **DID/去电号码活动**

如需展示其他报告，可参考 基于多条 SQL 查询的通话报告计算逻辑 了解各报告的数据逻辑，并按需添加 SQL 查询语句。

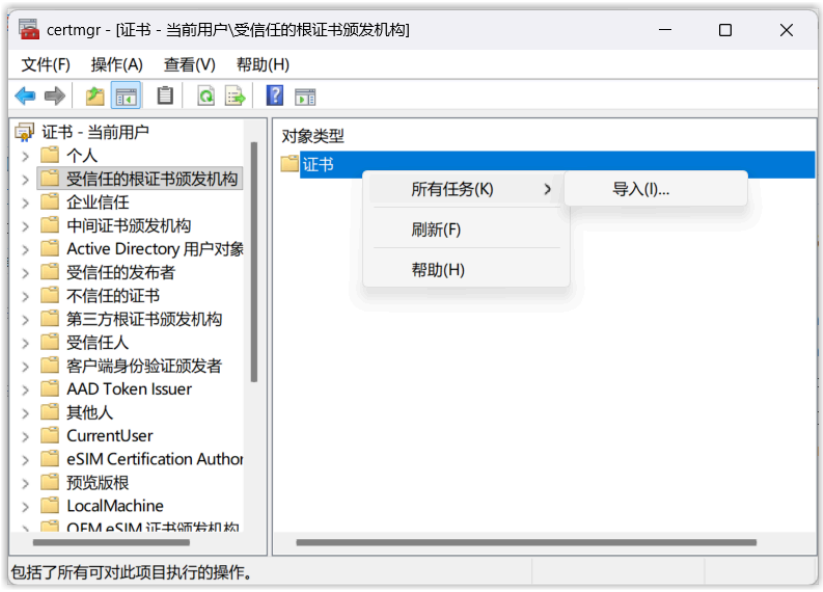
将 PBX 数据从 Microsoft SQL 导入 Power BI

前提条件

- 下载并解压 Power BI 模板。该压缩包包含适用于不同数据库的模板，请根据数据库类型选择合适的模板。
- 下载并安装 Power BI desktop。
- (可选) 如果 Microsoft SQL 服务器启用了 SSL，需安装证书。
    1. 下载 SSL 公共证书。
    2. 在受信任的根证书颁发机构存储中安装 SSL 证书。
        a. 在电脑上运行 `certmgr.msc`，打开证书管理器。



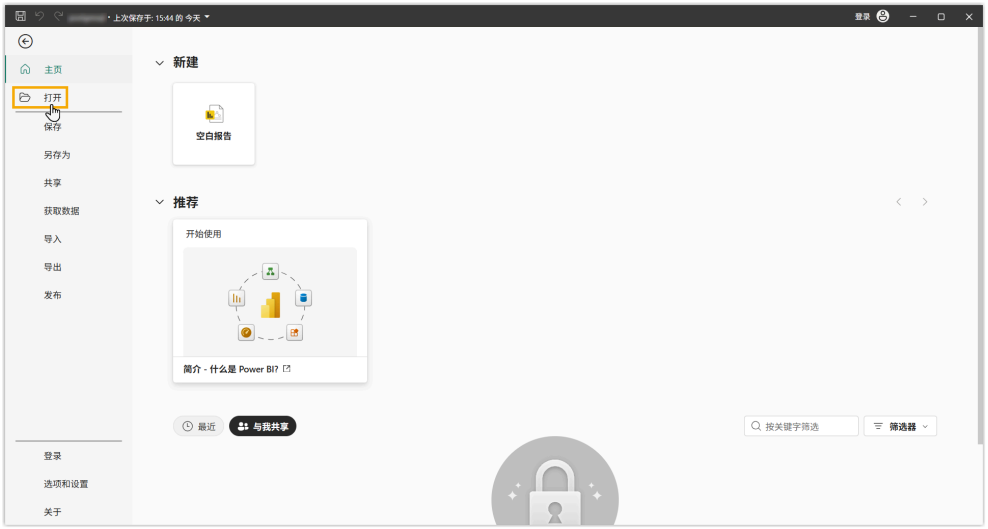
        b. 右键点击 **受信任的根证书颁发机构**，从 **所有任务** 列表中选择 **导入**。



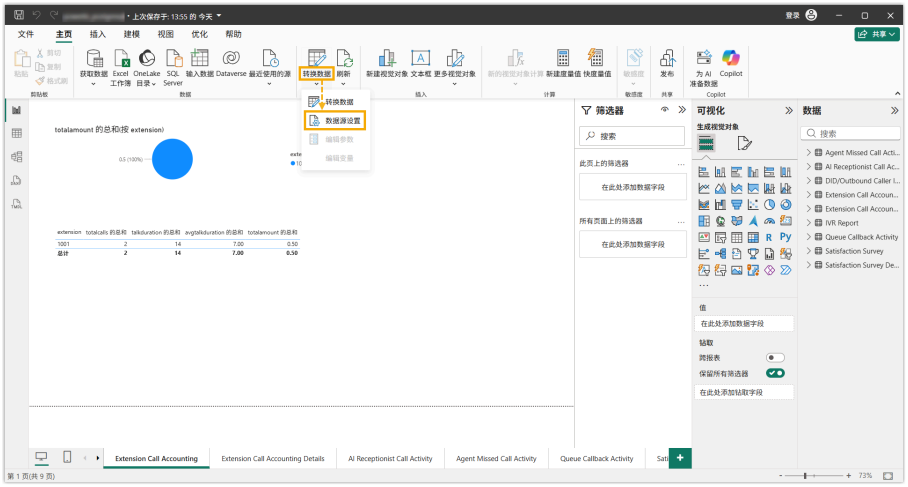
c. 按照向导中的提示导入根证书，然后选择 **确定**。

操作步骤

1. 启动 Power BI desktop。
2. 点击 **打开**，选择并打开 Yeastar 提供的模板文件。



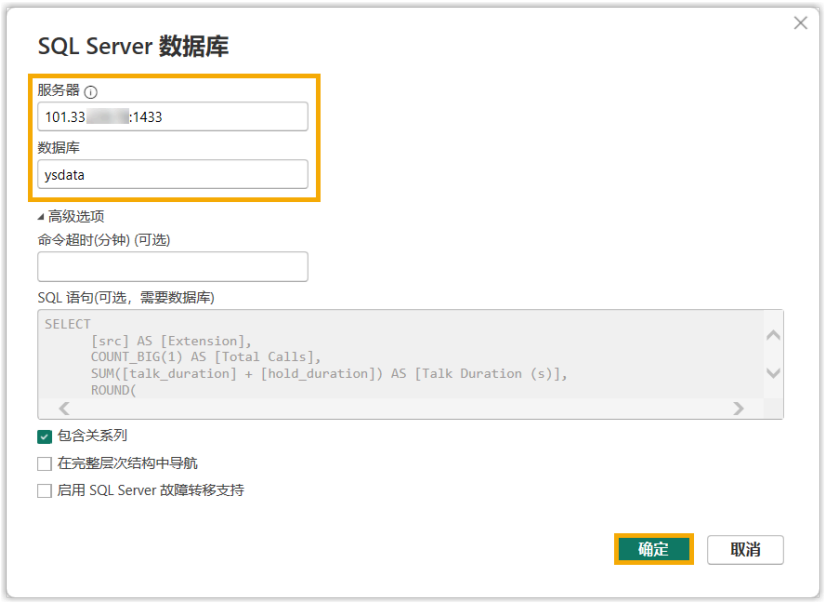
3. 更改数据源并配置访问权限。
    a. 在顶部工具栏，点击 **转换数据**，然后选择 **数据源设置**。



b. 更改数据源。

i. 选择默认的数据源，点击 **更改源**。

ii. 修改数据库连接信息，点击 **确定**。



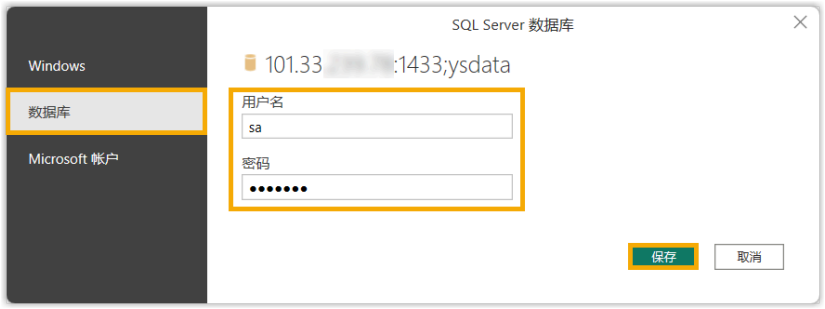
配置项	说明
服务器	填写数据库服务器的 IP 地址/域名及端口。
数据库	填写数据库名称。

c. 配置数据库访问凭证。

i. 点击 **编辑权限**。

ii. 在弹出的窗口中，点击 **编辑**。

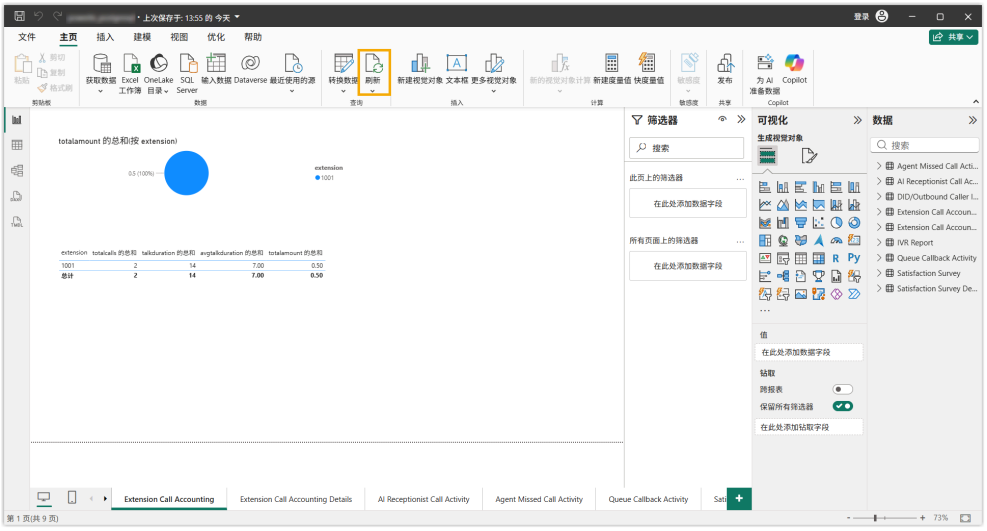
iii. 选择 **数据库**，填写认证信息，然后点击 **保存**。



配置项	说明
用户名	填写用于连接数据库的用户名。
密码	填写用于连接数据库的密码。

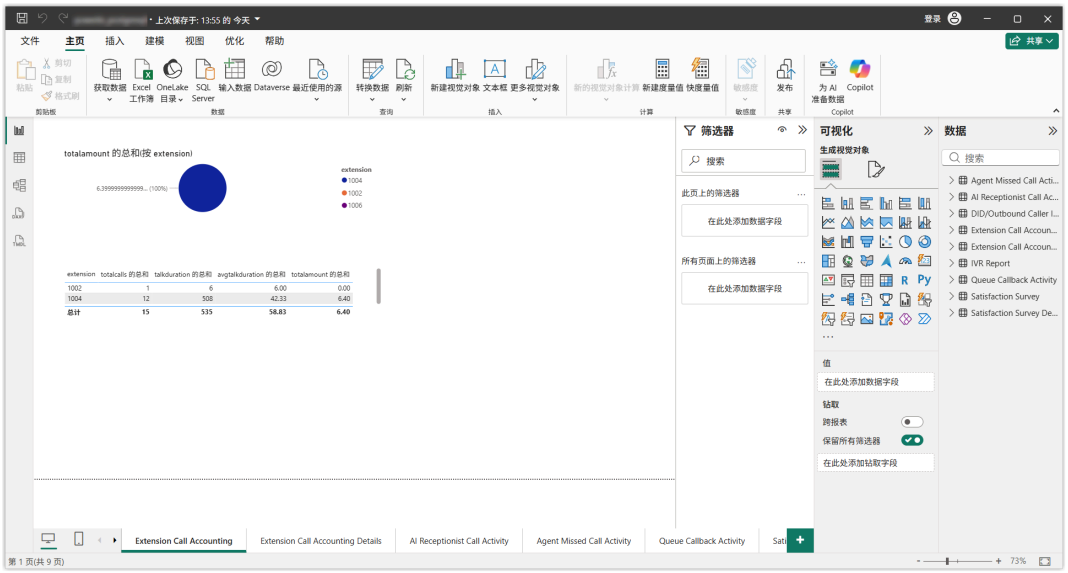
d. 保存设置。

4. 在页面顶部，点击 **刷新**，执行所有 SQL 查询。



执行结果

数据已成功导入 Power BI，实现数据可视化。



后续操作

按需添加或修改 SQL 查询语句，以自定义要展示的数据。

Note:
由于 Power BI 不支持在单个数据源中执行组合查询，因此 Yeastar 提供的仪表盘模板仅显示以下通话报告数据。

- **分机通话计费**

- **分机通话计费详情**
- **AI 接待员通话活动**
- **坐席未接通报告**
- **队列回拨报告**
- **满意度报告**
- **满意度调查详情**
- **IVR报告**
- **DID/去电号码活动**

如需展示其他报告，可参考 基于多条 SQL 查询的通话报告计算逻辑 了解各报告的数据逻辑，并按需添加 SQL 查询语句。

基于多条 SQL 查询的通话报告计算逻辑

由于 Grafana 和 Power BI 的功能限制，Yeastar 提供的模板仅支持使用单条 SQL 查询展示报告数据。本文提供参考信息，帮助你通过自定义 SQL 查询实现更多报告展示。

分机通话统计报告

要显示 **分机通话统计** 报告，需要分别查询每个分机的呼入和呼出通话统计数据，然后合并两部分查询结果，生成每个分机的完整通话统计信息。

1. 查询分机的呼出通话统计数据。

```
SELECT
  src as ext_num,
  SUM(failed) AS failed,
  SUM(vm) AS vm,
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN 1 WHEN c.call_flow = ''
 THEN c.answered WHEN c.call_flow != ''
    AND c.is_group = 1 THEN c.answered WHEN c.call_flow = 'Monitor'
 THEN c.answered ELSE 0 END
  ) AS answered,
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN 0 WHEN c.call_flow = ''
 THEN c.no_answer WHEN c.call_flow != ''
```

```
    AND c.is_group = 1 THEN c.no_answer WHEN c.call_flow = 'Monitor'
 THEN c.no_answer ELSE 0 END
  ) AS total_no_answered,
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN 0 WHEN c.call_flow = ''
 THEN c.busy WHEN c.call_flow != ''
    AND c.is_group = 1 THEN c.busy WHEN c.call_flow = 'Monitor' THEN
 c.busy ELSE 0 END
  ) AS busy,
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN 0 WHEN c.call_flow = ''
 THEN c.abandoned WHEN c.call_flow != ''
    AND c.is_group = 1 THEN c.abandoned WHEN c.call_flow = 'Monitor'
 THEN c.abandoned ELSE 0 END
  ) AS abandoned,
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN
 COALESCE(mobile.mobile_ring, 0) WHEN c.call_flow = '' THEN
 c.ring_duration WHEN c.call_flow != ''
    AND c.is_group = 1 THEN c.ring_duration WHEN c.call_flow =
 'Monitor' THEN c.ring_duration ELSE 0 END
  ) AS ring_duration,
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN
 COALESCE(mobile.mobile_talk, 0) WHEN c.call_flow = '' THEN (
      c.talk_duration + c.hold_duration
    ) WHEN c.call_flow != ''
    AND c.is_group = 1 THEN (
      c.talk_duration + c.hold_duration
    ) WHEN c.call_flow = 'Monitor' THEN (
      c.talk_duration + c.hold_duration
    ) ELSE 0 END
  ) AS talk_duration
FROM
  cdr.cdr_cleaned AS c
  LEFT JOIN (
    SELECT
      m.uid,
      MAX(m.answered) AS mobile_answered,
      MAX(
        CASE WHEN m.answered = 1 THEN m.ring_duration ELSE 0 END
```

```
      ) AS mobile_ring,
      MAX(
        CASE WHEN m.answered = 1 THEN m.talk_duration +
 m.hold_duration ELSE 0 END
      ) AS mobile_talk
    FROM
      cdr.cdr_cleaned AS m
    WHERE
      m.scenario = 'mobile'
    GROUP BY
      `m`.`uid`
  ) AS mobile ON c.uid = mobile.uid
WHERE
  c.src in (
    '1028', '1016', '1003', '1004', '1006',
    '1014', '1033', '1000', '1034', '1023',
    '1027', '1021', '1019', '1022', '1020',
    '1015', '1018', '1025', '1011', '1012',
    '1029', '1024', '1032', '1008', '1007',
    '1010', '1013', '1026', '1030', '1017',
    '1001', '1009', '1031', '1002', '1005'
  )
  AND c.start_ts >= 1773590400000000
  AND c.start_ts <= 1776268799999999
GROUP BY
  `src`
```

查询返回以下数据：

- `ext_num`：分机号码。
- `answered`：分机应答的来电总数。
- `total_no_answered`：分机未接的来电总数。
- `busy`：分机忙时的来电总数。
- `abandoned`：分机接听前呼叫者放弃的来电总数。
- `vm`：进入语音信箱的来电总数。
- `failed`：分机呼叫失败的去电总数。
- `ring_duration`：收到来电到来电被应答的时间。
- `talk_duration`：来电被应答到通话结束的时间。

2. 查询分机的呼入通话统计数据。

```
SELECT
  dst as ext_num,
  SUM(failed) AS failed,
  SUM(vm) AS vm,
```

```
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN 1 WHEN c.call_flow = ''
 THEN c.answered WHEN c.call_flow != ''
    AND c.is_group = 0 THEN c.answered ELSE 0 END
  ) AS answered,
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN 0 WHEN c.call_flow = ''
 THEN (c.no_answer) WHEN c.call_flow != ''
    AND c.is_group = 0
    AND c.is_display = 1 THEN (c.no_answer) ELSE 0 END
  ) AS total_no_answered,
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN 0 WHEN c.call_flow = ''
 THEN c.busy WHEN c.call_flow != ''
    AND c.is_group = 0
    AND c.is_display = 1 THEN c.busy ELSE 0 END
  ) AS busy,
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN 0 WHEN c.call_flow = ''
 THEN c.abandoned ELSE 0 END
  ) AS abandoned,
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN
 COALESCE(mobile.mobile_ring, 0) WHEN c.call_flow = '' THEN
 c.ring_duration WHEN c.call_flow != ''
    AND c.is_group = 0
    AND c.is_display = 1 THEN c.ring_duration ELSE 0 END
  ) AS ring_duration,
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN
 COALESCE(mobile.mobile_talk, 0) WHEN c.call_flow = '' THEN (
      c.talk_duration + c.hold_duration
    ) WHEN c.call_flow != ''
    AND c.is_group = 0 THEN (
      c.talk_duration + c.hold_duration
    ) ELSE 0 END
  ) AS talk_duration
FROM
  cdr.cdr_cleaned AS c
```

```
  LEFT JOIN (
    SELECT
      m.uid,
      MAX(m.answered) AS mobile_answered,
      MAX(
        CASE WHEN m.answered = 1 THEN m.ring_duration ELSE 0 END
      ) AS mobile_ring,
      MAX(
        CASE WHEN m.answered = 1 THEN m.talk_duration +
 m.hold_duration ELSE 0 END
      ) AS mobile_talk
    FROM
      cdr.cdr_cleaned AS m
    WHERE
      m.scenario = 'mobile'
    GROUP BY
      `m`.`uid`
  ) AS mobile ON c.uid = mobile.uid
WHERE
  c.dst in (
    '1028', '1016', '1003', '1004', '1006',
    '1014', '1033', '1000', '1034', '1023',
    '1027', '1021', '1019', '1022', '1020',
    '1015', '1018', '1025', '1011', '1012',
    '1029', '1024', '1032', '1008', '1007',
    '1010', '1013', '1026', '1030', '1017',
    '1001', '1009', '1031', '1002', '1005'
  )
  AND c.start_ts >= 1773590400000000
  AND c.start_ts <= 1776268799999999
GROUP BY
  `dst`
```

查询返回以下数据：

- `ext_num`：分机号码。
- `answered`：分机应答的来电总数。
- `total_no_answered`：分机未接的来电总数。
- `busy`：分机忙时的来电总数。
- `abandoned`：分机接听前呼叫者放弃的来电总数。
- `vm`：进入语音信箱的来电总数。
- `failed`：分机呼叫失败的去电总数。
- `ring_duration`：收到来电到来电被应答的时间。
- `talk_duration`：来电被应答到通话结束的时间。

3. 按分机号码合并呼入和呼出统计数据，生成完整的通话报告。

分机通话活动报告

要显示 **分机通话活动** 报告，需要分别查询每个分机的呼入和呼出通话统计数据，然后合并两部分查询结果，生成每个分机的完整通话统计信息。

1. 获取分机的呼出通话数据。

```
SELECT
  src as ext_num,
  MONTH(datetime) AS time,
  SUM(failed) AS failed,
  SUM(vm) AS vm,
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN 1 WHEN c.call_flow = ''
 THEN c.answered WHEN c.call_flow != ''
    AND c.is_group = 1 THEN c.answered WHEN c.call_flow = 'Monitor'
 THEN c.answered ELSE 0 END
  ) AS answered,
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN 0 WHEN c.call_flow = ''
 THEN c.no_answer WHEN c.call_flow != ''
    AND c.is_group = 1 THEN c.no_answer WHEN c.call_flow = 'Monitor'
 THEN c.no_answer ELSE 0 END
  ) AS total_no_answered,
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN 0 WHEN c.call_flow = ''
 THEN c.busy WHEN c.call_flow != ''
    AND c.is_group = 1 THEN c.busy WHEN c.call_flow = 'Monitor' THEN
 c.busy ELSE 0 END
  ) AS busy,
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN 0 WHEN c.call_flow = ''
 THEN c.abandoned WHEN c.call_flow != ''
    AND c.is_group = 1 THEN c.abandoned WHEN c.call_flow = 'Monitor'
 THEN c.abandoned ELSE 0 END
  ) AS abandoned,
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN
```

```
 COALESCE(mobile.mobile_ring, 0) WHEN c.call_flow = '' THEN
 c.ring_duration WHEN c.call_flow != ''
    AND c.is_group = 1 THEN c.ring_duration WHEN c.call_flow =
 'Monitor' THEN c.ring_duration ELSE 0 END
  ) AS ring_duration,
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN
 COALESCE(mobile.mobile_talk, 0) WHEN c.call_flow = '' THEN (
      c.talk_duration + c.hold_duration
    ) WHEN c.call_flow != ''
    AND c.is_group = 1 THEN (
      c.talk_duration + c.hold_duration
    ) WHEN c.call_flow = 'Monitor' THEN (
      c.talk_duration + c.hold_duration
    ) ELSE 0 END
  ) AS talk_duration
FROM
  cdr.cdr_cleaned AS c
  LEFT JOIN (
    SELECT
      m.uid,
      MAX(m.answered) AS mobile_answered,
      MAX(
        CASE WHEN m.answered = 1 THEN m.ring_duration ELSE 0 END
      ) AS mobile_ring,
      MAX(
        CASE WHEN m.answered = 1 THEN m.talk_duration +
 m.hold_duration ELSE 0 END
      ) AS mobile_talk
    FROM
      cdr.cdr_cleaned AS m
    WHERE
      m.scenario = 'mobile'
    GROUP BY
      `m`.`uid`
  ) AS mobile ON c.uid = mobile.uid
WHERE
  c.src in (
    '1032', '1008', '1007', '1010', '1013',
    '1026', '1030', '1017', '1001', '1009',
    '1031', '1002', '1005', '1028', '1016',
    '1003', '1004', '1006', '1014', '1033',
    '1000', '1034', '1023', '1027', '1021',
    '1019', '1022', '1020', '1015', '1018',
```

```
    '1025', '1011', '1012', '1029', '1024'
  )
  AND c.start_ts >= 1767196800000000
  AND c.start_ts <= 1798732799999999
GROUP BY
  src,
  time
ORDER BY
  time,
  ext_num asc
```

查询返回以下数据：

- `time`：时间。
- `ext_num`：分机号码。
- `answered`：分机应答的来电总数。
- `total_no_answered`：分机未接的来电总数。
- `busy`：分机忙时的来电总数。
- `abandoned`：分机接听前呼叫者放弃的来电总数。
- `failed`：分机呼叫失败的去电总数。
- `vm`：进入语音信箱的来电总数。
- `ring_duration`：收到来电到来电被应答的时间。
- `talk_duration`：来电被应答到通话结束的时间。

2. 获取分机的呼入通话数据。

```
SELECT
  dst as ext_num,
  MONTH(datetime) AS time,
  SUM(failed) AS failed,
  SUM(vm) AS vm,
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN 1 WHEN c.call_flow = ''
 THEN c.answered WHEN c.call_flow != ''
    AND c.is_group = 0 THEN c.answered ELSE 0 END
  ) AS answered,
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN 0 WHEN c.call_flow = ''
 THEN (c.no_answer) WHEN c.call_flow != ''
    AND c.is_group = 0
    AND c.is_display = 1 THEN (c.no_answer) ELSE 0 END
  ) AS total_no_answered,
  SUM(
```

```
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN 0 WHEN c.call_flow = ''
 THEN c.busy WHEN c.call_flow != ''
    AND c.is_group = 0
    AND c.is_display = 1 THEN c.busy ELSE 0 END
  ) AS busy,
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN 0 WHEN c.call_flow = ''
 THEN c.abandoned ELSE 0 END
  ) AS abandoned,
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN
 COALESCE(mobile.mobile_ring, 0) WHEN c.call_flow = '' THEN
 c.ring_duration WHEN c.call_flow != ''
    AND c.is_group = 0
    AND c.is_display = 1 THEN c.ring_duration ELSE 0 END
  ) AS ring_duration,
  SUM(
    CASE WHEN c.scenario = 'mobile' THEN 0 WHEN
 COALESCE(mobile.mobile_answered, 0) = 1 THEN
 COALESCE(mobile.mobile_talk, 0) WHEN c.call_flow = '' THEN (
      c.talk_duration + c.hold_duration
    ) WHEN c.call_flow != ''
    AND c.is_group = 0 THEN (
      c.talk_duration + c.hold_duration
    ) ELSE 0 END
  ) AS talk_duration
FROM
  cdr.cdr_cleaned AS c
  LEFT JOIN (
    SELECT
      m.uid,
      MAX(m.answered) AS mobile_answered,
      MAX(
        CASE WHEN m.answered = 1 THEN m.ring_duration ELSE 0 END
      ) AS mobile_ring,
      MAX(
        CASE WHEN m.answered = 1 THEN m.talk_duration +
 m.hold_duration ELSE 0 END
      ) AS mobile_talk
    FROM
      cdr.cdr_cleaned AS m
    WHERE
```

```
      m.scenario = 'mobile'
    GROUP BY
      `m`.`uid`
  ) AS mobile ON c.uid = mobile.uid
WHERE
  c.dst in (
    '1032', '1008', '1007', '1010', '1013',
    '1026', '1030', '1017', '1001', '1009',
    '1031', '1002', '1005', '1028', '1016',
    '1003', '1004', '1006', '1014', '1033',
    '1000', '1034', '1023', '1027', '1021',
    '1019', '1022', '1020', '1015', '1018',
    '1025', '1011', '1012', '1029', '1024'
  )
  AND c.start_ts >= 1767196800000000
  AND c.start_ts <= 1798732799999999
GROUP BY
  dst,
  time
ORDER BY
  time,
  ext_num asc
```

查询返回以下数据：

- `time`：时间。
- `ext_num`：分机号码。
- `answered`：分机应答的来电总数。
- `total_no_answered`：分机未接的来电总数。
- `busy`：分机忙时的来电总数。
- `abandoned`：分机接听前呼叫者放弃的来电总数。
- `failed`：分机呼叫失败的去电总数。
- `vm`：进入语音信箱的来电总数。
- `ring_duration`：收到来电到来电被应答的时间。
- `talk_duration`：来电被应答到通话结束的时间。

3. 按分机号码和时间合并呼入和呼出统计数据，生成完整的通话报告。

坐席呼叫摘要报告

要显示 **坐席呼叫摘要** 报告，需要分别查询每个坐席的呼入和呼出通话统计数据，合并结果后计算整体统计数据及平均通话时长。

1. 查询坐席的呼入队列通话统计数据。

```
SELECT
  c.dst AS agent,
  count(answered) AS answered_call,
  SUM(
    CASE WHEN sub.duration != NULL THEN c.duration + sub.duration ELSE
 c.duration END
  ) AS duration,
  SUM(talk_duration) AS talking_time,
  SUM(hold_duration) AS hold_time,
  SUM(
    CASE WHEN sub.ring_duration != NULL THEN c.ring_duration +
 sub.ring_duration ELSE c.ring_duration END
  ) AS waiting_time
FROM
  cdr.cdr_cleaned as c
  LEFT JOIN (
    SELECT
      uid,
      dst,
      sum(duration) as duration,
      sum(ring_duration) AS ring_duration
    FROM
      cdr.cdr_cleaned as c
    WHERE
      start_ts >= 1773590400000000
      AND start_ts <= 1776268799999999
      AND (
        c.dst IN ('1000', '1001')
      )
      AND src != 'PBX'
      AND call_flow_number = '6404'
      AND call_flow = 'Queue'
      AND disposition != 'ANSWERED'
      AND is_group = '0'
    GROUP BY
      uid,
      dst
  ) AS sub ON c.uid = sub.uid
  and c.dst = sub.dst
WHERE
  start_ts >= 1773590400000000
  AND start_ts <= 1776268799999999
  AND (
    c.dst IN ('1000', '1001')
  )
```

```
  AND src != 'PBX'
  AND call_flow_number = '6404'
  AND call_flow = 'Queue'
  AND disposition = 'ANSWERED'
  AND is_group = '0'
GROUP BY
  `c`.`dst`
```

查询返回以下数据：

- `agent`：坐席号码。
- `answered_call`：坐席应答的队列来电数量。
- `duration`：坐席处理队列来电的时间，从响铃开始至通话结束。
- `talking_time`：坐席处理队列来电的通话时长。
- `hold_time`：坐席保持队列来电的时长。
- `waiting_time`：队列来电被接听前的等待时长。

2. 获取坐席的呼出队列通话统计数据。

```
SELECT
  src AS agent,
  COUNT(uid) AS total_call,
  SUM(answered) AS answered_call,
  SUM(
    CASE WHEN disposition = 'ANSWERED' THEN duration ELSE 0 END
  ) AS duration,
  SUM(
    CASE WHEN disposition = 'ANSWERED' THEN talk_duration ELSE 0 END
  ) AS talking_time,
  SUM(
    CASE WHEN disposition = 'ANSWERED' THEN hold_duration ELSE 0 END
  ) AS hold_time,
  SUM(
    CASE WHEN disposition = 'ANSWERED' THEN ring_duration ELSE 0 END
  ) AS waiting_time
FROM
  `cdr`.`cdr_cleaned`
WHERE
  start_ts >= 1773590400000000
  AND start_ts <= 1776268799999999
  AND call_type = 'Outbound'
  AND src in ('1000', '1001')
  AND NOT(srcname = 'Conference Call')
  AND is_group = '0'
GROUP BY
  `src`
```

查询返回以下数据：

- `agent`：坐席号码。
- `total_call`：坐席发起的呼出通话总数。
- `answered_call`：坐席发起且被应答的呼出通话总数。
- `duration`：坐席呼出通话时长，从响铃开始到通话结束。
- `talking_time`：坐席呼出通话时长，从接通开始到通话结束。
- `hold_time`：坐席在呼出通话中的保持时长。
- `waiting_time`：坐席呼出通话在被接听前的等待时长。

3. 按坐席分机号码合并呼入和呼出通话统计数据，生成整体统计信息。

- `total_duration`：坐席呼入和呼出通话的 `duration` 总时长。
- `total_talk_time`：坐席呼入和呼出通话中 `talking_time` 与 `hold_time` 的总时长。
- `total_hold_time`：坐席呼入和呼出通话的 `hold_time` 总时长。
- `total_waiting_time`：坐席呼入和呼出通话的 `waiting_time` 总时长。
- `total_call`：坐席呼入通话的 `answered_call` 与坐席呼出通话的 `total_call` 总数之和。
- `inbound_answered_call`：坐席接听的通话总数。
- `inbound_duration`：坐席呼入队列通话中 `talking_time` 与 `hold_time` 的总时长。
- `outbound_total_call`：坐席呼出队列通话总数。
- `outbound_answered_call`： 坐席呼出队列通话中已接听的通话总数。
- `outbound_duration`：坐席呼出队列通话中 `talking_time` 与 `hold_time` 的总时长。

4. 计算每个坐席的平均时长。

- `avg_server_time` = `total_duration` / (`inbound_answered_call` + `outbound_answered_call`)
- `avg_talking_time` = `total_talk_time` / (`inbound_answered_call` + `outbound_answered_call`)
- `avg_hold_time` = `total_hold_time` / (`inbound_answered_call` + `outbound_answered_call`)
- `avg_waiting_time` = `total_waiting_time` /(`inbound_answered_call`+ `outbound_answered_call`)

坐席签入报告

要显示 **坐席签入报告**，需要从队列日志中获取坐席的登录和登出事件记录。

```
SELECT
```

```
  agent,
  timestamp,
  event
FROM
  `queue_log`
WHERE
  (
    queuename = 'queue-6402'
    and event in('ADDMEMBER', 'REMOVEMEMBER')
  )
  AND (timestamp >= 1773590400)
  AND (timestamp <= 1776268799)
GROUP BY
  agent,
  timestamp,
  event
ORDER BY
  timestamp asc
```

查询返回以下数据：

- `agent`：坐席号码。
- `timestamp`：坐席签入或签出队列的时间。
- `event`：
  - `ADDMEMBER`：坐席签入队列。
  - `REMOVEMEMBER`：坐席签出队列。

> **Note:**
> 总登录时长按队列日志中 `REMOVEMEMBER` 和 `ADDMEMBER` 事件的时间戳差计算。

坐席暂停报告

要显示 **坐席暂停报告**，需要从队列日志中获取坐席的暂停和恢复事件记录。

```
SELECT
  agent,
  timestamp,
  event,
  data1
FROM
  `queue_log`
WHERE
  (
```

```
    queuename = 'queue-6404'
    and event in(
      'PAUSE', 'UNPAUSE', 'REMOVEMEMBER'
    )
  )
  AND (timestamp >= 1774022400)
  AND (timestamp <= 1776700799)
GROUP BY
  agent,
  timestamp,
  event,
  data1
ORDER BY
  timestamp asc
```

查询返回以下数据：

- `agent`：坐席号码。
- `timestamp`：坐席暂停或恢复服务的时间。
- `data1`：暂停原因。
- `event`：
  - `PAUSE`：坐席暂停服务。
  - `UNPAUSE`：坐席恢复服务。

Note:
总暂停时长按 `UNPAUSE` 和 `PAUSE` 事件的时间戳差计算。

坐席业绩报告

要显示 **坐席业绩** 报告，需要从多个数据集汇总队列通话数据，并合并结果以计算最终的坐席业绩指标。

1. 汇总不包含 `q_half_consult` 数据的队列业绩数据。

```
SELECT
  `dst`,
  Count(*) as total_calls,
  sum(answered) as answered_calls,
  sum(missed) as missed_calls,
  sum(abandoned) as abandoned_calls,
  max(ring_duration) as max_ring_time,
  sum(answered_ring_duration) as total_answered_ring_time,
```

```
  sum(ring_duration) as total_ring_time,
  sum(hold_duration) as total_hold_time,
  sum(talk_duration) as total_talking_time,
  sum(member_hold_duration) as total_member_hold_duration,
  sum(member_answered_count) as total_member_answered
FROM
  cdr.cdr_cleaned as c
WHERE
  c.is_group = 1
  AND (
    not (
      c.ring_duration <= 1
      and c.abandoned = 1
    )
  )
  AND flag != 'q_half_consult'
  AND c.start_ts >= 1776614400000000
  AND c.start_ts <= 1776700799999999
  AND call_flow = 'Queue'
  AND call_flow_number = '6404'
GROUP BY
  `dst`
```

查询返回以下数据：

- `dst`：队列号码。
- `total_calls`：队列收到的来电数量。
- `answered_calls`：队列接听的通话数量。
- `missed_calls`：队列未接听的通话数量。
- `abandoned_calls`：呼叫者在接通坐席前放弃的通话数量。
- `max_ring_time`：呼叫者在队列中等待并被坐席接通前的最长等待时间。
- `total_answered_ring_time`：已接听通话中，从通话开始到被接听的时间。
- `total_ring_time`：从通话开始到被接听的时间。
- `total_hold_time`：通话保持的总时长。
- `total_talking_time`：从通话接通到通话结束的时间。
- `total_member_hold_duration`：坐席保持通话的时长。
- `total_member_answered`：坐席接听的通话数量。

2. 汇总 `q_half_consult` 相关的队列业绩数据。

```
SELECT
  c.dst,
  sum(c.ring_duration) as total_ring_time,
  sum(
    CASE WHEN h.answered > 0 THEN c.ring_duration ELSE 0 END
```

```
  ) as total_answered_ring_time,
  max(
    c.ring_duration + h.ring_duration
  ) as max_ring_time
FROM
  cdr.cdr_cleaned as c
  LEFT JOIN (
    SELECT
      uid,
      answered,
      ring_duration
    FROM
      cdr.cdr_cleaned as c
    WHERE
      c.is_group = 1
      AND flag != 'q_half_consult'
      AND c.start_ts >= 1776614400000000
      AND c.start_ts <= 1776700799999999
      AND call_flow = 'Queue'
      AND call_flow_number = '6404'
  ) AS h ON h.uid = c.another_uid
WHERE
  c.is_group = 1
  AND flag = 'q_half_consult'
  AND c.start_ts >= 1776614400000000
  AND c.start_ts <= 1776700799999999
  AND call_flow = 'Queue'
  AND call_flow_number = '6404'
GROUP BY
  `c`.`dst`
```

查询返回以下数据：

- `dst`：队列号码。
- `total_ring_time`：通话开始到被接听的时间。
- `total_answered_ring_time`：已接听通话中，从通话开始到被接听的时间。
- `max_ring_time`：呼叫者在队列中等待并被坐席接通前的最长等待时间。

3. 合并两个数据集中的队列业绩数据，生成最终的队列统计结果。
   - `total_answered_ring_time` = 两个数据集中 `total_answered_ring_time` 的总和
   - `total_ring_time` = 两个数据集中 `total_ring_time` 的总和
   - `max_ring_time` = 两个数据集中 `max_ring_time` 的最大值
4. 基于汇总后的数据计算队列业绩指标。
   - `average_waiting_time` = `total_answered_ring_time` / `answered_calls`

- `average_talking_time` = (`total_talking_time` + `total_hold_time`) / `answered_calls`
- `average_handle_time` = (`total_answered_ring_time` + `total_talking_time` + `total_hold_time`) / `answered_calls`
- `average_hold_time` = `total_hold_time` / `answered_calls`
- `anwsered_rate` = `answered_calls` / `total_calls`
- `missed_rate` = `missed_calls` / `total_calls`
- `abandoned_rate` = `abandoned_calls` / `total_calls`
- `all_call_average_waiting_time` = `total_ring_time` / `total_calls`

5. 基于坐席维度汇总队列通话数据 (排除回拨相关通话及无效放弃通话)。

```
SELECT
  c.uid,
  c.call_flow_number as queue,
  c.dst as agent,
  sum(c.answered) as answered_calls,
  sum(c.missed) as missed_calls,
  max(c.ring_duration) as max_ring_time,
  sum(c.answered_ring_duration) as total_answered_ring_time,
  sum(c.ring_duration) as total_ring_time,
  sum(c.talk_duration) as total_talking_time,
  sum(c.hold_duration) as total_hold_time
FROM
  cdr.cdr_cleaned as c
  LEFT JOIN cdr.cdr_cleaned AS h ON h.uid = c.uid
  AND h.is_group = 1
  AND h.ring_duration <= 1
  AND h.disposition = 'ABANDONED'
WHERE
  c.call_flow IN ('Queue', 'QCB')
  AND c.call_flow_number = '6404'
  AND c.dst IN ('1000', '1001')
  AND c.is_group != 1
  AND c.disposition != 'ABANDONED'
  AND c.src != 'PBX'
  AND h.uid IS NULL
  AND c.start_ts >= 1776614400000000
  AND c.start_ts <= 1776700799999999
GROUP BY
  c.call_flow_number,
  c.uid,
  c.dst
```

查询返回以下数据:

- `uid`：通话记录唯一 ID。
- `queue`：队列号码。
- `agent`：坐席号码。
- `answered_calls`：坐席接听的通话数量。
- `missed_calls`：坐席未接听的来电数量。
- `max_ring_time`：坐席接听前的最大响铃时间。
- `total_answered_ring_time`：已接听通话中，从通话开始到接通的时间。
- `total_ring_time`：从通话开始到接通的时间。
- `total_talking_time`：从接通到通话结束的时间。
- `total_hold_time`：坐席保持通话的时长。

6. 基于坐席维度汇总回拨相关队列通话数据。

```
SELECT
  c.uid,
  c.call_flow_number as queue,
  c.dst as agent,
  sum(c.answered) as answered_calls,
  sum(c.missed) as missed_calls,
  max(c.ring_duration) as max_ring_time,
  sum(c.answered_ring_duration) as total_answered_ring_time,
  sum(c.ring_duration) as total_ring_time,
  sum(c.talk_duration) as total_talking_time,
  sum(c.hold_duration) as total_hold_time
FROM
  cdr.cdr_cleaned as c
  LEFT JOIN cdr.cdr_cleaned AS h ON h.uid = c.uid
  AND h.is_group = 1
  AND h.ring_duration <= 1
  AND h.disposition = 'ABANDONED'
WHERE
  c.call_flow IN ('Queue', 'QCB')
  AND c.call_flow_number = '6404'
  AND c.dst IN ('1000', '1001')
  AND c.is_group != 1
  AND c.disposition != 'ABANDONED'
  AND c.src = 'PBX'
  AND h.uid IS NULL
  AND c.start_ts >= 1776614400000000
  AND c.start_ts <= 1776700799999999
GROUP BY
  c.call_flow_number,
  c.uid,
  c.dst
```

查询返回以下数据：

- `uid`：通话记录唯一 ID。
- `queue`：队列号码。
- `agent`：坐席号码。
- `answered_calls`：坐席接听的通话数量。
- `missed_calls`：坐席未接听的来电数量。
- `max_ring_time`：坐席接听前的最大响铃时间。
- `total_answered_ring_time`：已接听通话从通话开始到通话被应答的时长。
- `total_ring_time`：通话开始到通话被应答的时长。
- `total_talking_time`：通话接通到通话结束的时间。
- `total_hold_time`：坐席保持通话的时长。

7. 基于坐席维度获取回拨统计数据。

```
SELECT
  call_flow_number as queue,
  src as agent,
  count(*) as total_calls,
  sum(answered) as answered_calls,
  sum(missed) as missed_calls,
  max(ring_duration) as max_ring_time,
  sum(answered_ring_duration) as total_answered_ring_time,
  sum(ring_duration) as total_ring_time,
  sum(duration) as total_talking_time,
  sum(hold_duration) as total_hold_time
FROM
  `cdr`.`cdr_cleaned`
WHERE
  call_flow_number = '6404'
  AND is_qcb = 1
  AND src IN ('1000', '1001')
  AND start_ts >= 1776614400000000
  AND start_ts <= 1776700799999999
  AND call_flow = ('Queue')
GROUP BY
  call_flow_number,
  src
```

查询返回以下数据：

- `queue`：队列号码。
- `agent`：坐席号码。
- `total_calls`：收到的来电数量。
- `answered_calls`：坐席接听的通话数量。

- `missed_calls`：坐席未接听的来电数量。
- `max_ring_time`：坐席接听前的最大响铃时间。
- `total_answered_ring_time`：已接听通话从通话开始到通话被应答的时长。
- `total_ring_time`：通话开始到通话被应答的时长。
- `total_talking_time`：通话接通到通话结束的时间。
- `total_hold_time`：坐席保持通话的总时长。

8. 将回拨通话的通话时长计入坐席统计数据。

```
agent_talk_time_map[queue_agent]=total_talking_time
```

9. 基于队列、坐席维度汇总通话数据。

```
agentMap[queue_agent].dst=agent
agentMap[queue_agent].total_calls+=answered_calls+missed_calls
agentMap[queue_agent].answered_calls+=answered_calls
agentMap[queue_agent].missed_calls+=missed_calls
agentMap[queue_agent].total_ring_time+=total_ring_time

agentMap[queue_agent].total_talking_time+=agent_talk_time_map[queue_ag
ent]//只加一次
agent_answered_ring_duration[queue_uid_agent] =
 total_answered_ring_time
if answered_calls > 0 {
    agent_answered[queue_uid_agent] = 1
}
```

10. 计算坐席业绩指标。

```
 //
answered_ring_time = total_answered_ring_time
if answered_calls > 0 || agent_answered[queue_uid_agent] == 1 {
    answered_ring_time = total_ring_time
}
agentMap.total_calls += answered_calls + missed_calls
agentMap.answered_calls += answered_calls
agentMap.missed_calls += missed_calls
agentMap.total_answered_ring_time += answered_ring_time
agentMap.total_ring_time += total_ring_time
agentMap.total_talking_time += total_talking_time + total_hold_time
//
agent_answered_ring_duration[queue_uid_agent] += answered_ring_time

//
agentMap.max_ring_time =
 max(agent_answered_ring_duration[queue_agent]...)
```

```
//
agentMap.avg_talking_time = agentMap.total_talking_time /
 agentMap.answered_calls
agentMap.avg_waiting_time = agentMap.total_answered_ring_time /
 agentMap.answered_calls
agentMap.miss_rate=agentMap.missed_calls / agentMap.total_calls
```

队列平均等待和通话时间

要显示 **队列平均等待和通话时间** 报告，需要对队列通话数据进行汇总，并基于汇总结果计算相关指标。

1. 从队列通话记录中排除 `q_half_consult` 数据，构建基础数据集 `normal_data`。

```
SELECT
  DAY(datetime) AS time,
  Count(*) as total_calls,
  sum(answered) as answered_calls,
  sum(answered_ring_duration) as total_answered_ring_time,
  sum(ring_duration) as total_ring_time,
  sum(talk_duration + hold_duration) as total_talking_time
FROM
  `cdr`.`cdr_cleaned`
WHERE
  is_group = 1
  AND flag != 'q_half_consult'
  AND start_ts >= 1774972800000000
  AND start_ts <= 1777564799999999
  AND call_flow = 'Queue'
  AND call_flow_number = '6404'
GROUP BY
  `time`
```

   查询返回以下数据：

   - `time`：收到来电的时间。
   - `total_calls`：队列收到的来电总数。
   - `answered_calls`：队列接听的来电总数。
   - `total_answered_ring_time`：已接听通话从通话开始到通话被应答的时长。
   - `total_ring_time`：通话开始到通话被应答的时长。
   - `total_talking_time`：通话被应答到通话结束的时长。

2. 从队列通话记录中筛选 `q_half_consult` 数据，构建数据集 `q_half_consult_data`。

```
SELECT
  day AS time,
```

```
  sum(c.ring_duration) as total_ring_time,
  sum(
    CASE WHEN h.answered > 0 THEN c.ring_duration ELSE 0 END
  ) as total_answered_ring_time,
  max(
    c.ring_duration + h.ring_duration
  ) as max_ring_time
FROM
  cdr.cdr_cleaned as c
  LEFT JOIN (
    SELECT
      uid,
      answered,
      ring_duration
    FROM
      cdr.cdr_cleaned as c
    WHERE
      c.is_group = 1
      AND flag != 'q_half_consult'
      AND c.start_ts >= 1774972800000000
      AND c.start_ts <= 1777564799999999
      AND call_flow = 'Queue'
      AND call_flow_number = '6404'
  ) AS h ON h.uid = c.another_uid
WHERE
  is_group = 1
  AND flag = 'q_half_consult'
  AND start_ts >= 1774972800000000
  AND start_ts <= 1777564799999999
  AND call_flow = 'Queue'
  AND call_flow_number = '6404'
GROUP BY
  `time`
```

查询返回以下数据：

- `time`：收到来电的时间。
- `total_ring_time`：通话开始到通话被应答的时长。
- `total_answered_ring_time`：已接听通话从通话开始到通话被应答的时长。
- `max_ring_time`：呼叫者在队列中等待的最长时间。

3. 将 `normal_data` 与 `q_half_consult_data` 进行汇总，得到最终指标结果。

```
nornal_data[time].time
nornal_data[time].total_calls
nornal_data[time].answered_calls
nornal_data[time].total_talking_time
```

```
nornal_data[time].total_answered_ring_time +=
 q_half_consult_data[time].total_answered_ring_time
nornal_data[time].total_ring_time+=
 q_half_consult_data[time].total_ring_time
nornal_data[time].average_waiting_time =
 nornal_data[time].total_answered_ring_time /
 nornal_data[time].answered_calls
nornal_data[time].average_talking_time
 = nornal_data[time].total_taking_time /
 nornal_data[time].answered_calls
nornal_data[time].all_call_average_waiting_time =
 nornal_data[time].total_ring_time/ nornal_data[time].total_calls
```

队列回拨摘要报告

要显示 **队列回拨摘要** 报告，需要基于回拨相关字段对队列通话记录进行汇总统计。

```
SELECT
  dst,
  COUNT(*) AS total_count,
  SUM(CASE WHEN is_qcb = 1 THEN 1 ELSE 0 END) AS qcb_count,
  SUM(
    CASE WHEN is_qcb = 1
    AND qcb_status = 'success' THEN 1 ELSE 0 END
  ) AS qcb_success_count,
  SUM(
    CASE WHEN is_qcb = 1
    AND qcb_status != 'success' THEN 1 ELSE 0 END
  ) AS qcb_failed_count
FROM
  `cdr`.`cdr_cleaned`
WHERE
  start_ts >= 1774108800000000
  AND start_ts <= 1776787199999999
  AND call_flow_number in ('6404')
  AND call_flow = 'Queue'
  AND is_group = '1'
GROUP BY
  `dst`
```

查询返回以下数据：

- `dst`：队列号码。
- `qcb_failed_count`：队列回拨失败的总数。
- `qcb_success_count`：队列成功回拨的总数。

- `qcb_count`：呼叫者成功请求队列回拨的总数。
- `total_count`：队列接收的来电总数。

队列业绩报告

要展示 **队列业绩** 报告，需要对队列通话数据进行汇总，并基于汇总结果计算相关指标。

1. 从队列通话记录中排除 `q_half_consult` 数据，构建基础数据集 `normal_data`。

```
SELECT
  `dst`,
  Count(*) as total_calls,
  sum(answered) as answered_calls,
  sum(missed) as missed_calls,
  sum(abandoned) as abandoned_calls,
  max(ring_duration) as max_ring_time,
  sum(answered_ring_duration) as total_answered_ring_time,
  sum(ring_duration) as total_ring_time,
  sum(hold_duration) as total_hold_time,
  sum(talk_duration) as total_talking_time,
  sum(member_hold_duration) as total_member_hold_duration,
  sum(member_answered_count) as total_member_answered,
  SUM(
    CASE WHEN dst = '6404'
    AND disposition = 'ANSWERED'
    AND (
      ring_duration - COALESCE(join_announcement_duration, 0)
    )< 60 THEN 1 ELSE 0 END
  ) AS total_sla_call
FROM
  cdr.cdr_cleaned as c
WHERE
  c.is_group = 1
  AND (
    not (
      c.ring_duration <= 1
      and c.abandoned = 1
    )
  )
  AND (
    not (
      c.talk_duration < 1
      and c.answered = 1
    )
  )
```

```
  AND flag != 'q_half_consult'
  AND c.start_ts >= 1774108800000000
  AND c.start_ts <= 1776787199999999
  AND call_flow = 'Queue'
  AND call_flow_number = '6404'
GROUP BY
  `dst`
```

2. 从队列通话记录中筛选 `q_half_consult` 数据，构建数据集 `q_half_consult_data`。

```
SELECT
  c.dst,
  sum(c.abandoned) as abandoned_calls,
  sum(c.ring_duration) as total_ring_time,
  sum(
    CASE WHEN h.answered > 0 THEN c.ring_duration ELSE 0 END
  ) as total_answered_ring_time,
  max(
    c.ring_duration + h.ring_duration
  ) as max_ring_time
FROM
  cdr.cdr_cleaned as c
  LEFT JOIN (
    SELECT
      uid,
      answered,
      ring_duration
    FROM
      cdr.cdr_cleaned as c
    WHERE
      c.is_group = 1
      AND flag != 'q_half_consult'
      AND c.start_ts >= 1774108800000000
      AND c.start_ts <= 1776787199999999
      AND call_flow = 'Queue'
      AND call_flow_number = '6404'
  ) AS h ON h.uid = c.another_uid
WHERE
  c.is_group = 1
  AND flag = 'q_half_consult'
  AND c.start_ts >= 1774108800000000
  AND c.start_ts <= 1776787199999999
  AND call_flow = 'Queue'
  AND call_flow_number = '6404'
GROUP BY
  `c`.`dst`
```

3. 将 `normal_data` 与 `q_half_consult_data` 进行汇总，得到最终指标结果。

```
nornal_data[dst].queue_num = nornal_data[dst].dst
nornal_data[dst].total_calls = nornal_data[dst].total_calls
nornal_data[dst].answered_calls = nornal_data[dst].answered_calls
nornal_data[dst].abandoned_calls = nornal_data[dst].abandoned_calls
nornal_data[dst].missed_calls = nornal_data[dst].missed_calls
nornal_data[dst].total_sla_call = nornal_data[dst].total_sla_call
nornal_data[dst].total_answered_ring_time +=
 q_half_consult_data[dst].total_answered_ring_time
nornal_data[dst].total_ring_time +=
 q_half_consult_data[dst].total_ring_time
if q_half_consult_data[dst].max_ring_time >
 nornal_data[dst].max_ring_time {
    nornal_data[dst].max_waiting_time =
 q_half_consult_data[dst].max_ring_time
}
nornal_data[dst].average_waiting_time=nornal_data[dst].total_answered_
ring_time/nornal_data[dst].answered_calls
nornal_data[dst].average_talking_time=(nornal_data[dst].total_talking_
time+nornal_data[dst].total_hold_time)/nornal_data[dst].answered_calls
nornal_data[dst].average_handle_time=(nornal_data[dst].total_answered_
ring_time+nornal_data[dst].total_talking_time+nornal_data[dst].total_h
old_time)/nornal_data[dst].answered_calls
nornal_data[dst].average_hold_time=(nornal_data[dst].total_hold_time)
/nornal_data[dst].answered_calls

nornal_data[dst].anwsered_rate=nornal_data[dst].answered_calls/nornal_
data[dst].total_calls
nornal_data[dst].missed_rate=nornal_data[dst].missed_calls/nornal_data
[dst].total_calls
nornal_data[dst].abandoned_rate=nornal_data[dst].abandoned_calls/norna
l_data[dst].total_calls
nornal_data[dst].all_call_average_waiting_time=nornal_data[dst].total_
ring_time/nornal_data[dst].total_calls
```

队列业绩活动报告

要展示 **队列业绩活动** 报告，需要对队列通话数据进行汇总，并基于汇总结果计算相关指标。

1. 从队列通话记录中排除 `q_half_consult` 数据，构建基础数据集 `normal_data`。

```
SELECT
  day AS time,
  Count(*) as total_calls,
```

```
  sum(answered) as answered_calls,
  sum(missed) as missed_calls,
  sum(abandoned) as abandoned_calls,
  max(ring_duration) as max_ring_time,
  sum(answered_ring_duration) as total_answered_ring_time,
  sum(ring_duration) as total_ring_time,
  sum(hold_duration) as total_hold_time,
  sum(talk_duration) as total_talking_time,
  sum(member_hold_duration) as total_member_hold_duration,
  sum(member_answered_count) as total_member_answered,
  SUM(
    CASE WHEN dst = '6404'
    AND disposition = 'ANSWERED'
    AND (
      ring_duration - COALESCE(join_announcement_duration, 0)
    )< 60 THEN 1 ELSE 0 END
  ) AS total_sla_call
FROM
  `cdr`.`cdr_cleaned`
WHERE
  is_group = 1
  AND flag != 'q_half_consult'
  AND (
    not (
      ring_duration <= 1
      and abandoned = 1
    )
  )
  AND start_ts >= 1774972800000000
  AND start_ts <= 1777564799999999
  AND call_flow = 'Queue'
  AND call_flow_number = '6404'
GROUP BY
  `time`
```

2. 从队列通话记录中筛选 `q_half_consult` 数据，构建数据集 `q_half_consult_data`。

```
SELECT
  day AS time,
  sum(c.abandoned) as abandoned_calls,
  sum(c.ring_duration) as total_ring_time,
  sum(
    CASE WHEN h.answered > 0 THEN c.ring_duration ELSE 0 END
  ) as total_answered_ring_time,
  max(
    c.ring_duration + h.ring_duration
```

```
  ) as max_ring_time
FROM
  cdr.cdr_cleaned as c
  LEFT JOIN (
    SELECT
      uid,
      answered,
      ring_duration
    FROM
      cdr.cdr_cleaned as c
    WHERE
      c.is_group = 1
      AND flag != 'q_half_consult'
      AND c.start_ts >= 1774972800000000
      AND c.start_ts <= 1777564799999999
      AND call_flow = 'Queue'
      AND call_flow_number = '6404'
  ) AS h ON h.uid = c.another_uid
WHERE
  is_group = 1
  AND flag = 'q_half_consult'
  AND start_ts >= 1774972800000000
  AND start_ts <= 1777564799999999
  AND call_flow = 'Queue'
  AND call_flow_number = '6404'
GROUP BY
  `time`
```

3. 将 `normal_data` 与 `q_half_consult_data` 进行汇总，得到最终指标结果。

```
nornal_data[time].time = nornal_data[dst].time
nornal_data[time].total_calls = nornal_data[time].total_calls
nornal_data[time].answered_calls = nornal_data[time].answered_calls
nornal_data[time].abandoned_calls = nornal_data[time].abandoned_calls
nornal_data[time].missed_calls = nornal_data[time].missed_calls
nornal_data[time].total_sla_call = nornal_data[time].total_sla_call
nornal_data[time].total_answered_ring_time +=
 q_half_consult_data[time].total_answered_ring_time
nornal_data[time].total_ring_time +=
 q_half_consult_data[time].total_ring_time
if q_half_consult_data[time].max_ring_time >
 nornal_data[time].max_ring_time {
    nornal_data[time].max_waiting_time =
 q_half_consult_data[time].max_ring_time
}
```

```
nornal_data[time].answered_hold_time =
 nornal_data[time].total_hold_time
nornal_data[time].answered_call_time =
 nornal_data[time].total_answered_ring_time+nornal_data[time].total_ta
lking_time+nornal_data[time].total_hold_time
nornal_data[time].total_talk_time =
 nornal_data[time].total_talking_time+nornal_data[time].total_hold_tim
e
nornal_data[time].answered_waiting_time =
 nornal_data[time].total_answered_ring_time
nornal_data[time].average_waiting_time=nornal_data[time].total_answere
d_ring_time/nornal_data[time].answered_calls
nornal_data[time].average_talking_time=(nornal_data[time].total_talkin
g_time+nornal_data[time].total_hold_time)/nornal_data[time].answered_c
alls
nornal_data[time].average_handle_time=(nornal_data[time].total_answere
d_ring_time+nornal_data[time].total_talking_time+nornal_data[time].tot
al_hold_time)/nornal_data[time].answered_calls
nornal_data[time].average_hold_time=(nornal_data[time].total_hold_time
)/nornal_data[time].answered_calls
nornal_data[time].total_waiting_time =
 nornal_data[time].total_ring_time
nornal_data[time].sla = nornal_data[time].total_sla_call /
 nornal_data[time].total_calls
nornal_data[time].anwsered_rate=nornal_data[time].answered_calls/norna
l_data[time].total_calls
nornal_data[time].missed_rate=nornal_data[time].missed_calls/nornal_da
ta[time].total_calls
nornal_data[time].abandoned_rate=nornal_data[time].abandoned_calls/nor
nal_data[time].total_calls
nornal_data[time].all_call_average_waiting_time=nornal_data[time].tota
l_ring_time/nornal_data[time].total_calls
```

未回电报告

要显示 **未回电报告**，需要查询未接来电数据，并结合相关的接听记录与回拨记录进行分析，以判断每条未接来电是否已被回拨。

1. 查询未接来电数据。

```
SELECT
  id,
  uid,
  timestamp,
  src,
```

```
  srcname,
  dst,
  dstname,
  ring_duration,
  dstnameprefix,
  disposition,
  call_flow,
  call_type
FROM
  `cdr`.`cdr_cleaned`
WHERE
  call_type = 'Inbound'
  AND is_display = 1
  AND disposition in ('ABANDONED', 'NO ANSWER', 'BUSY')
  AND (
    NOT(
      disposition = 'ABANDONED'
      and ring_duration < 5
    )
  )
  AND start_ts >= 1774108800000000
  AND start_ts <= 1776787199999999
ORDER BY
  uid,
  timestamp;
```

2. 按 `uid` 对未接来电数据进行去重，保留每个 `uid` 最新的一条记录，构建基础数据集 `list_map`。

```
list_map[uid].uid = miss_call_data.uid
list_map[uid].timestamp = miss_call_data.timestamp
list_map[uid].src = miss_call_data.src
list_map[uid].srcname = miss_call_data.srcname
list_map[uid].dst= miss_call_data.dst
list_map[uid].dstname = miss_call_data.dstname
list_map[uid].call_to_type = miss_call_data.call_flow
list_map[uid].unreturn_status = 0
list_map[uid].return_time = 0
list_map[uid].miss_call_type = miss_call_data.disposition
list_map[uid].call_type = miss_call_data.call_type
list_map[uid].ring_duration = miss_call_data.ring_duration
```

3. 查询与主叫号码相关的接听记录。

```
SELECT
  max(timestamp) as timestamp,
  src,
```

```
  dst,
  disposition
FROM
  `cdr`.`cdr_cleaned`
WHERE
  timestamp > 1775534367893043
  AND is_display = 1
  AND disposition = 'ANSWERED'
  AND (
    src = '2233123456'
    or dst = '2233123456'
  )
GROUP BY
  src,
  dst,
  disposition
```

4. 统计主叫号码对应的最新接听时间。

```
//has_return_time_map
if answered_call_data[src].timestamp > has_return_time_map[src] {
    has_return_time_map[src] = answered_call_data[src].timestamp
}
```

5. 获取与未接来电主叫号码相关的回拨记录。

```
SELECT
  max(timestamp) as timestamp,
  src,
  dst,
  disposition
FROM
  `cdr`.`cdr_cleaned`
WHERE
  timestamp >= 1775534367893043
  AND is_display = 1
  AND (
    (dst = '2233123456')
    or (
      src = '2233123456'
      and disposition = 'ANSWERED'
    )
  )
GROUP BY
  src,
  dst,
  disposition
```

6. 判断每个未接来电主叫号码对应的最新回拨时间。

```
//last_return_time_map
if last_answered_call_data[src].timestamp > last_return_time_map[src]
 {
    last_return_time_map[src] = last_answered_call_data[src].timestamp
}
```

7. 判断每个未接来电的回电状态。

```
list_map[uid].unreturn_status=2
src = list_map[uid].src
//
if has_return_time_map[src] > list_map[uid].timestamp {
    list_map[uid].unreturn_status = 1
}
//
if last_return_time_map[src] > list_map[uid].timestamp {
    list_map[uid].return_time = last_return_time_map[src]
}
```

响铃组统计报告

要显示 **响铃组统计** 报告，需要分别从响铃组维度和成员维度汇总通话数据，并计算通话统计指标，得到最终指标结果。

1. 从响铃组维度汇总通话统计数据，生成 list 数据集。

```
SELECT
  dst as ring_group_num,
  SUM(
    CASE WHEN disposition = 'ANSWERED' THEN 1 ELSE 0 END
  ) AS answered_calls,
  count(*) as total_calls
FROM
  `cdr`.`cdr_cleaned`
WHERE
  is_group = 1
  AND start_ts >= 1776441600000000
  AND start_ts <= 1779119999999999
  AND call_flow = 'RingGroup'
  AND call_flow_number in ('6300', '6301', '6302')
GROUP BY
  `dst`
```

2. 从成员维度汇总通话统计数据，生成 member_list 数据集。

```
SELECT
  `dst`,
  call_flow_number as ring_group_num,
  SUM(
    CASE WHEN disposition = 'ANSWERED' THEN 1 ELSE 0 END
  ) AS answered_calls,
  count(*) as total_calls
FROM
  `cdr`.`cdr_cleaned`
WHERE
  is_group = 0
  AND start_ts >= 1776441600000000
  AND start_ts <= 1779119999999999
  AND call_flow = 'RingGroup'
  AND call_flow_number in ('6300', '6301', '6302')
GROUP BY
  dst,
  call_flow_number
```

3. 按接通率对 list 和 member_list 进行降序排序，生成最终统计结果。

```
{"ring_group_num":"6200","answered_calls":10,"total_calls":20,"member_
list":[{"ext_num":"1000","answered_calls":10,"total_calls":20}...]}
```

PBX 通话活动报告

要显示 **PBX 通话活动** 报告，需要从三类数据源汇总通话数据：中继通话、设备通话以及内部通话。各部分数据分别计算后，汇总用于整体通话分析。

1. 获取通过中继发起或接收的通话数据。
    a. 获取中继通话数据，不包含多方通话。

```
SELECT
  max(day) AS time,
  max(srctrunk) as srctrunk,
  max(dsttrunk) as dsttrunk,
  SUM(talk_duration + hold_duration) as total_talk_duration,
  MAX(concurrent) as max_concurrent_call,
  uid,
  0 as total_calls
FROM
  `cdr`.`cdr_cleaned`
WHERE
  (
    srctrunk in (
```

```
      '97账号中继注册段238', '235点对点kk'
    )
    or dsttrunk in (
      '97账号中继注册段238', '235点对点kk'
    )
  )
  AND is_display = 1
  AND (
    NOT(
      dstnameprefix in ('Queue', 'RingGroup')
      and disposition = 'ANSWERED'
    )
  )
  AND NOT(srcname = 'Conference Call')
  AND start_ts >= 1774972800000000
  AND start_ts <= 1777564799999999
GROUP BY
  `uid`;
```

b. 计算同一中继在相同时间范围内的通话数、总通话时长及最大通话并发数 (不包含多方通话)。

c. 获取中继通话数据，包含多方通话。

```
SELECT
  max(day) AS time,
  srctrunk,
  dsttrunk,
  SUM(talk_duration + hold_duration) as total_talk_duration,
  MAX(concurrent) as max_concurrent_call,
  uid,
  COUNT(*) as total_calls
FROM
  `cdr`.`cdr_cleaned`
WHERE
  (
    srctrunk in (
      '97账号中继注册段238', '235点对点kk'
    )
    or dsttrunk in (
      '97账号中继注册段238', '235点对点kk'
    )
  )
  AND is_display = 1
  AND (
    NOT(
      dstnameprefix in ('Queue', 'RingGroup')
```

```
      and disposition = 'ANSWERED'
    )
  )
  AND srcname = 'Conference Call'
  AND start_ts >= 1774972800000000
  AND start_ts <= 1777564799999999
GROUP BY
  srctrunk,
  dsttrunk,
  uid;
```

d. 计算同一中继在相同时间范围内的通话数、总通话时长及最大通话并发数 (包含多方通话)。

2. 获取系统通话数据，包括内部通话、所有中继的呼入与呼出通话等。

a. 获取设备通话数据，不包含语音会议和广播组通话。

```
SELECT
  max(day) AS time,
  uid,
  SUM(talk_duration + hold_duration) as total_talk_duration,
  MAX(concurrent) as max_concurrent_call
FROM
  `cdr`.`cdr_cleaned`
WHERE
  (
    NOT(
      dstnameprefix in ('Queue', 'RingGroup')
      and disposition = 'ANSWERED'
    )
  )
  AND call_flow != 'Audio Conference'
  AND call_flow != 'Paging'
  AND dstnameprefix != 'Paging'
  AND start_ts >= 1774972800000000
  AND start_ts <= 1777564799999999
GROUP BY
  `uid`;
```

b. 计算同一时间范围内设备通话的通话数、总通话时长及最大通话并发数 (不包含多方通话)。

c. 查询语音会议与广播组通话数据。

```
SELECT
  uid,
  src,
  dst,
```

```
  timestamp,
  talk_duration,
  hold_duration,
  concurrent,
  dstnameprefix,
  call_flow,
  call_type,
  month,
  day,
  hour
FROM
  `cdr`.`cdr_cleaned`
WHERE
  (
    NOT(
      dstnameprefix in ('Queue', 'RingGroup')
      and disposition = 'ANSWERED'
    )
  )
  AND (
    call_flow = 'Audio Conference'
    OR call_flow = 'Paging'
    OR dstnameprefix = 'Paging'
  )
  AND start_ts >= 1774972800000000
  AND start_ts <= 1777564799999999
ORDER BY
  timestamp asc,
  dstnameprefix desc
LIMIT
  5000;
```

d. 应用以下规则：

- **广播组通话**：相同 `uid` 视为一条通话记录；设备通话数按记录逐条累加；通话时长仅取广播组记录 (`dstnameprefix = Paging` 对应的 `talk_duration` 和 `hold_duration` 加和)；最大并发数基于广播组记录计算。
- **语音会议**：相同 `uid_src_dst` 视为一条通话记录；设备通话数按记录逐条累加；通话时长为 `talk_duration` 和 `hold_duration` 加和；最大并发数基于汇总后的语音会议数据计算。

e. 计算同一时间范围内设备通话的通话数、总通话时长及最大通话并发数 (包含多方通话)。

3. 获取内部通话数据。

a. 获取内部通话数据，不包含语音会议和广播组通话。

```
SELECT
  max(day) AS time,
  uid,
  SUM(talk_duration + hold_duration) as total_talk_duration,
  MAX(concurrent) as max_concurrent_call
FROM
  `cdr`.`cdr_cleaned`
WHERE
  call_type = 'Internal'
  AND src != 'PBX'
  AND (
    NOT(
      dstnameprefix in ('Queue', 'RingGroup')
      and disposition = 'ANSWERED'
    )
  )
  AND call_flow != 'Audio Conference'
  AND call_flow != 'Paging'
  AND dstnameprefix != 'Paging'
  AND start_ts >= 1774972800000000
  AND start_ts <= 1777564799999999
GROUP BY
  `uid`;
```

b. 计算同一时间范围内内部通话的通话数、总通话时长及最大通话并发数 (不包含多方通话)。

c. 查询语音会议与广播组通话数据。

```
SELECT
  uid,
  src,
  dst,
  timestamp,
  talk_duration,
  hold_duration,
  concurrent,
  dstnameprefix,
  call_flow,
  call_type,
  month,
  day,
  hour
FROM
  `cdr`.`cdr_cleaned`
WHERE
```

```
  call_type = 'Internal'
  AND (
    NOT(
      dstnameprefix in ('Queue', 'RingGroup')
      and disposition = 'ANSWERED'
    )
  )
  AND (
    call_flow = 'Audio Conference'
    OR call_flow = 'Paging'
    OR dstnameprefix = 'Paging'
  )
  AND start_ts >= 1774972800000000
  AND start_ts <= 1777564799999999
ORDER BY
  timestamp asc,
  dstnameprefix desc
LIMIT
  5000;
```

d. 应用以下规则：

- **广播组通话**：相同 `uid` 视为一条通话记录；内部通话数按记录逐条累加；通话时长仅取广播组记录 (`dstnameprefix = Paging` 对应的 `talk_duration` 和 `hold_duration` 加和)；最大并发数基于广播组记录计算。
- **语音会议**：相同 `uid_src_dst` 视为一条通话记录；内部通话数按记录逐条累加；通话时长为 `talk_duration` 和 `hold_duration` 加和；最大并发数基于汇总后的语音会议数据计算。

e. 计算同一时间范围内内部通话的通话数、总通话时长及最大通话并发数 (包含多方通话)。

转写使用量详情

要显示 **转录使用量详情** 报告，你需要按使用类型统计转录总时长。

```
SELECT
  usage_type,
  SUM(duration) as total_duration
FROM
  `pbx`.`ai_usage_record`
WHERE
  (
    usage_type in (
      'call_transcription', 'voicemail_transcription'
    )
```

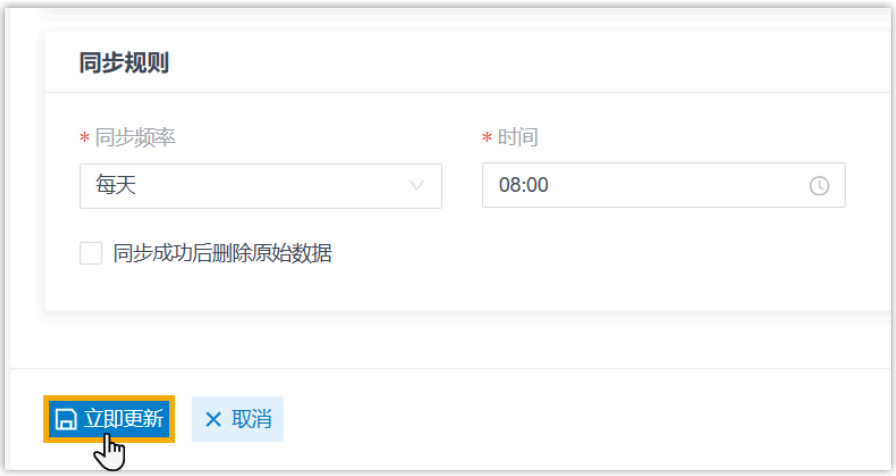
```
  )
  AND (timestamp >= 1767196800)
  AND (timestamp < 1779206399)
GROUP BY
  usage_type
```

手动触发数据同步到第三方数据库

默认情况下，Yeastar P 系列云 PBX 会按照配置的同步周期将指定数据同步至第三方数据库。你也可以随时手动触发同步，上次同步后产生的新数据将立即同步至数据库。

操作步骤

1. 登录 PBX 管理网页，进入 **应用对接 > 数据连接器**。
2. 手动触发数据同步。
   a. 在页面底部，点击 **立即更新**。



   b. 在弹出的窗口中，点击 **确定**。

      系统将校验数据库连接状态。
   c. 在弹出的窗口中，点击 **确认**。

执行结果

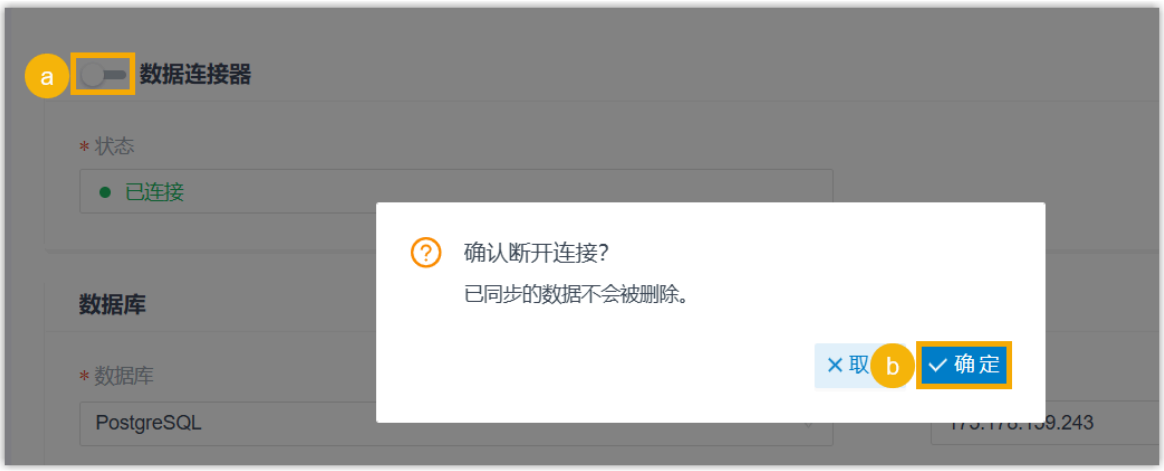
上次同步后产生的新数据将立即同步至数据库。

禁用数据连接器集成

本文介绍如何在 Yeastar P 系列云 PBX 上禁用数据连接器集成。

操作步骤

1. 登录 PBX 管理网页，进入 **应用对接 > 数据连接器**。
2. 禁用数据连接器集成。



   a. 在集成页面顶部，关闭开关。
   b. 在弹出的窗口中，点击 **确定**。

执行结果

- **状态** 显示为 “断开连接”，表示集成已断开。
- 已同步的数据将保留在第三方数据库中，不会被删除。