Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

Колледж информатики и программирования

Специальность 09.02.03 Программирование в компьютерных системах

**ОТЧЕТ**

**ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

**(по профилю специальности)**

Профессиональный модульПМ.02 Разработка и администрирование баз данных

*(наименование профессионального модуля)*

Выполнил:

обучающийся учебной группы № 3ПКС-120

Веремюк Е. А.

*(И.О. Фамилия)*

Проверил:

руководитель практики от колледжа:

Сибирев И.В.

*(И.О. Фамилия)*

**Москва**

**20­­­­23**

Содержание

[**Работа с BI** 4](#_Toc139059389)

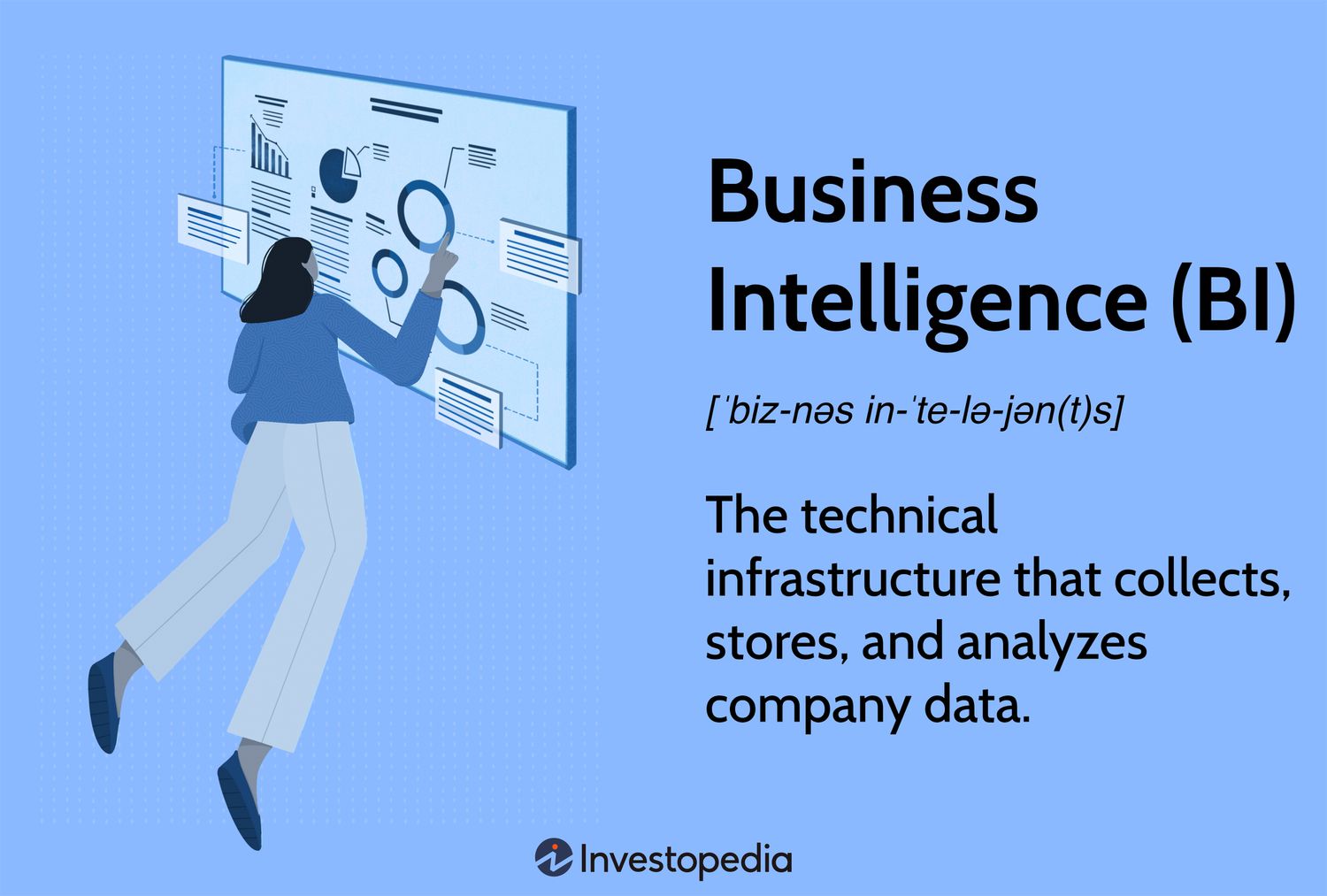
[**Ход работы** 5](#_Toc139059390)

[**Как работает сборщик данных Microsoft SQL** 8](#_Toc139059392)

[**Вывод** 12](#_Toc139059393)

# **Работа с BI**

Цель работы: Изучить тему использования BI-технологий для анализа и визуализации данных с целью принятия обоснованных и эффективных управленческих решений, улучшения бизнес-процессов и повышения конкурентоспособности организации.

  
  
Задачи работы с технологиями BI:

1. Сбор данных: Собрать необходимые данные из различных источников, таких как базы данных, веб-сервисы, файлы Excel и другие источники.

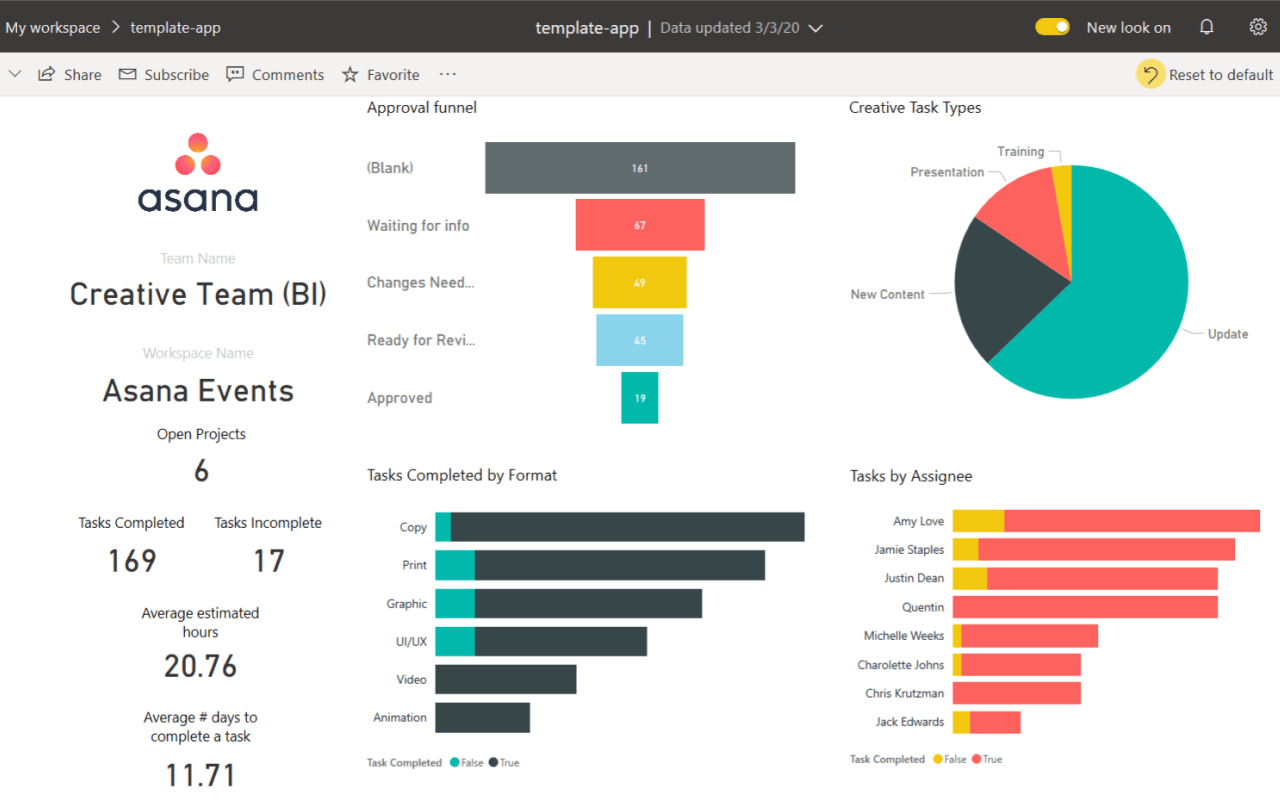
2. Чистка и трансформация данных: Проанализировать и очистить данные от ошибок, дубликатов и пропусков. Применить необходимые преобразования данных для их структурирования и готовности к анализу и визуализации.

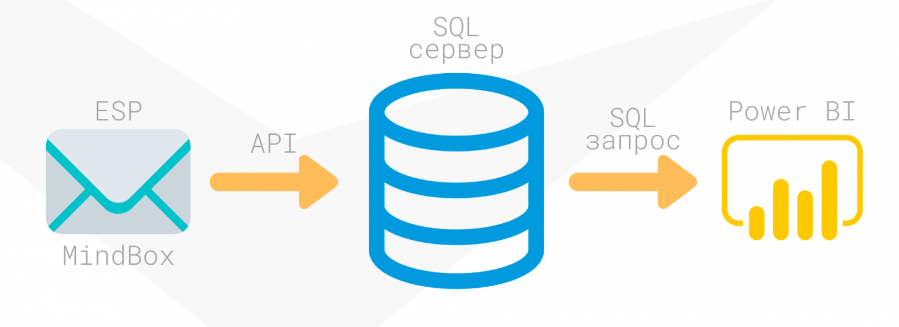
3. Моделирование и анализ данных: Создать модели данных и провести анализ данных для выявления трендов, паттернов и взаимосвязей между различными переменными. Использовать статистические методы и алгоритмы для получения полезных insights.

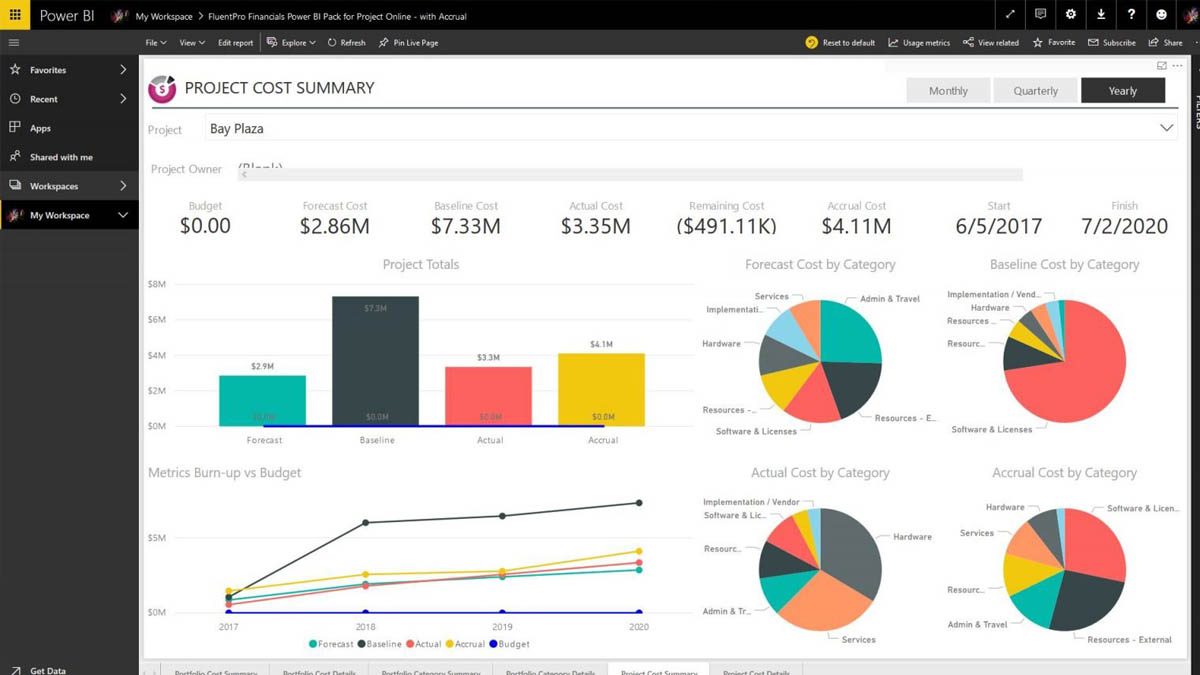
4. Создание отчетов и дашбордов: Разработать отчеты и дашборды, которые предоставят понятную и наглядную информацию о ключевых метриках и показателях производительности бизнеса. Отчеты и дашборды должны быть интерактивными и гибкими для быстрого и удобного анализа данных.

# **Ход работы**

BI (Business Intelligence) - это совокупность процессов, методов и технологий, направленных на сбор, хранение, анализ, представление и предоставление данных, используемых для поддержки принятия управленческих решений в организации.



Главная цель BI - предоставить пользователям организации доступ к актуальной, достоверной и полной информации, которая поможет им принимать обоснованные решения, улучшать эффективность бизнес-процессов, оптимизировать ресурсы и повысить конкурентоспособность компании.  
  
Основные компоненты BI включают:  
  
1. ETL (Extract, Transform, Load) - процесс извлечения данных из различных источников, их преобразования и загрузки в целевую систему хранения данных (Data Warehouse или Data Mart).  
  
2. Data Warehouse (DWH) или хранилище данных - это централизованная база данных, в которой собраны и хранятся данные из разных источников. DWH обычно строится в виде звездчатой схемы или схемы "Снежинки".  
  
3. OLAP (Online Analytical Processing) - технология анализа данных, позволяющая пользователям проводить многомерный анализ данных, строить отчеты, агрегировать данные и отображать их в удобной для восприятия форме. OLAP позволяет проводить анализ данных по различным измерениям и иерархиям.  
  
4. Data Mining - технология анализа данных, позволяющая автоматически искать скрытые закономерности, тренды и неочевидные взаимосвязи в больших объемах данных. Data Mining используется для выявления новых знаний и прогнозирования будущих событий.

5. Reporting and Dashboards - инструменты и технологии для создания и представления отчетов и панелей управления, которые позволяют пользователю визуализировать данные, контролировать бизнес-показатели и быстро принимать решения на основе актуальной информации.  
  
BI помогает управленческим работникам и аналитикам получать доступ к нужной информации в удобной форме, проводить анализ данных, выявлять тенденции и тренды, прогнозировать результаты и принимать эффективные бизнес-решения. Это позволяет компании быть более гибкой, адаптивной и конкурентоспособной на рынке.

## К примеру, мы можем импортировать данные о действиях, связанных с письмами, отправленными по разным сегментам, из Mindbox. В данном случае действие представляет собой последовательность событий, начиная от отправки письма и заканчивая попаданием в спам. При написании таких скриптов следует обратить внимание на следующие важные, но не всегда очевидные моменты. Mindbox предоставляет данные постранично, а не одним потоком. Максимальное количество записей на один запрос составляет 10 000, но лучше получать данные пакетами по 500-1000 записей, чтобы снизить вероятность возникновения ошибок в API. Поэтому для каждого сегмента потребуется выполнить несколько запросов. Важно сохранить последовательность этих запросов так, чтобы данные оставались в том же порядке, что и в Mindbox. API Mindbox позволяет получать данные с применением фильтров. Важно фильтровать данные с учетом уникального идентификатора действия (ID), чтобы получить только те действия, ID которых больше начального значения, если мы собираем данные за определенный период. Скрипт может выполняться длительное время, поэтому важно создать механизм для дополнения уже собранных данных. Таким образом, можно ежедневно добавлять новые данные, а не собирать все заново.

## **Как работает сборщик данных Microsoft SQL**

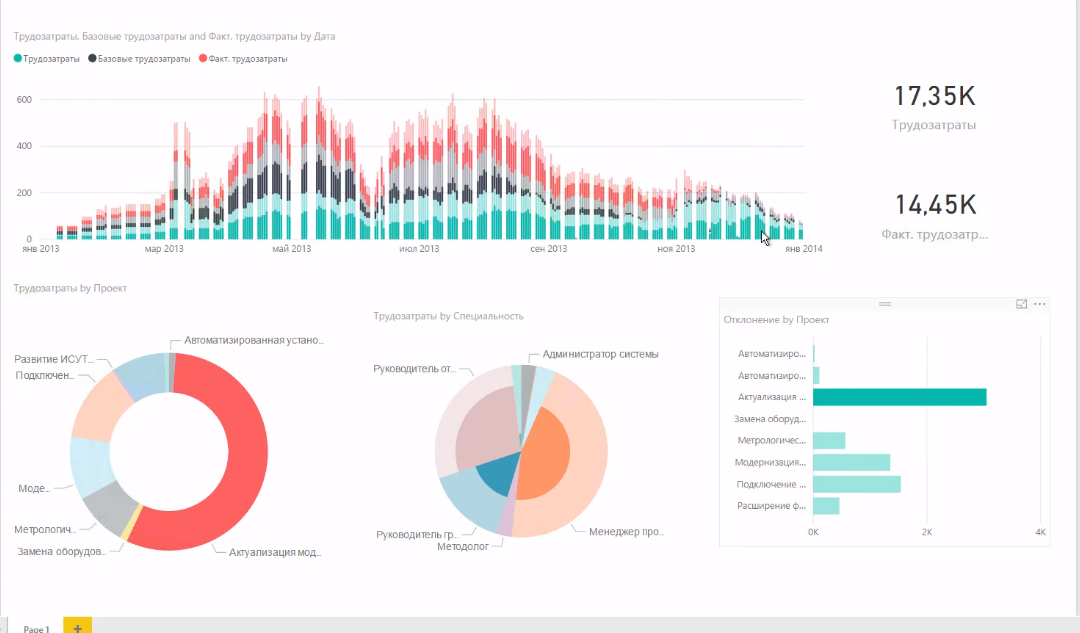
Алгоритм программы можно описать примерно так:

1. Узнать максимальный ID действия, который мы успели сохранить в нашей базе.
2. Получить через API Mindbox действия, ID которых больше, чем максимальный ID в нашей базе. Если данных пока нет, то начинать с нулевого ID.
3. Если данные получены, то записываем их в нашу базу данных. Возвращаемся к пункту 1. Если же данных нет, значит, мы либо собрали их все раньше, либо Mindbox вернул ошибку.

Пример листинга:

const config = require('./config.js');  
  
Ожидаемое содержимое конфигурационного файла:  
module.exports = {  
  
mssql: {  
user: 'sa',  
password: 'mssqlpassword',  
server: 'mssqlhost',  
database: 'mssqldbname',  
connectionTimeout: 60000,  
requestTimeout: 60000  
},  
  
mindbox\_api\_key: 'ключ\_доступа\_к\_API\_mindbox'  
};  
\*\*/  
  
const async = require('async');  
const request = require('request');  
const https = require("https");  
const sql = require('mssql');  
var parseString = require('xml2js').parseString;  
var \_ = require('lodash');  
var iconv = require('iconv-lite');  
  
//Подключение к SQL-серверу  
sql.connect(config.mssql, err => {  
// Query  
if (err) {  
console.log('SQL connect error: ', err, config.mssql);  
}  
})  
  
sql.on('error', err => {  
console.log('SQL server error: ', err);  
// ... error handler  
})  
  
setTimeout(function() {  
//Будем испортировать данные по нескольким операциям, сохраняя данные в одноименные таблицы  
var operarions = ['EksportDejstvijPoRuchnym', 'EksportDejstvijPoRassylkam', 'EksportDejstvijPoRassylkamSkandinavskij', 'EksportDejstvijPoRassylkamNovoe', 'EksportDejstvijPoRassylkamSkolkovskij', 'EksportDejstvijPoRassylkamDsk1', 'EksportDejstvijPoRassylkamDyxanie', 'EksportDejstvijPoRassylkamPokolenie', 'EksportDejstvijPoRassylkamFsk'];  
async.forEach(operarions, function(operation, cb1) {  
mbxChunk(operation, cb1);  
},  
function(err, res) {  
if (err) {  
sql.close();  
process.exit();  
return;  
}  
console.log('DONE!');  
sql.close();  
process.exit();  
}  
)  
}, 1000);  
  
/\*\*  
\* Рекурсивная функция получения данных и записи в таблицу  
\*/  
function mbxChunk(operation, cb) {  
console.log('New Chunk '+operation);  
var sqlreq = new sql.Request();  
//Получаем максимальный mindboxId из таблицы, чтобы знать, с какого ID собирать данные  
sqlreq.query("SELECT MAX(mindboxId) AS last FROM "+config.mssql.database+".dbo."+operation+";", function(err, maxres) {  
if (err || !maxres.recordset[0]) {  
console.log('MsSQL SELECT MAX error: ', err);  
cb(err);  
return;  
}  
var chunk = parseInt(maxres.recordset[0].last || 0) + 1;  
  
//Получим данные, начиная с ID=chunk  
mbxCall(operation, chunk, function(err, xml) {  
console.log(operation + ' from #' + chunk.toString() + ' started');  
if (err) {  
console.log(operation + ' from #' + chunk.toString() + ': Mindbox API request error: ', err);  
cb(err);  
}  
parseString(xml, function(err, res) {  
if (err) {  
console.log(operation + ' from #' + chunk.toString() + ': Mindbox returns invalid XML, parse error: ', err);  
console.log(xml);  
cb(err);  
}  
if (res && res.result && res.result.status == 'Success') {  
var rows = res.result.customerActions ? res.result.customerActions[0].customerAction : [];  
if (rows.length > 0) {  
mbxHandle(operation, rows, function(err, res) {  
console.log(operation + ' from #' + chunk.toString() + ' + '+rows.length.toString()+' done!');  
mbxChunkCopy(operation, cb);  
});  
}  
else {  
console.log(operation + ' - data is over!');  
cb();  
}  
}  
else {  
console.log(operation + ' from #' + chunk.toString() + ': Mindbox returns unsuccessfull result: ', res);  
cb(err);  
}  
});  
})  
})  
}  
  
function mbxChunkCopy(operation, cb) {  
mbxChunk(operation, cb);  
}  
  
/\*\*  
\* Запрос к API Mindbox  
\* Получаем 1000 записей. ID записи начинается со startId  
\*/  
function mbxCall(operation, startId, cb) {  
var request = require("request");  
var options = {  
method: 'POST',  
url: '[https://api.mindbox.ru/v3/operations/sync'](https://vk.com/away.php?utf=1&to=https%3A%2F%2Fapi.mindbox.ru%2Fv3%2Foperations%2Fsync%27),  
qs: {  
endpointId: 'fskuniversalpoint', operation: operation  
},  
headers: {  
"Cache-Control": "no-cache",  
"Content-Type": "application/xml",  
"Accept": "application/xml",  
"Authorization": 'Mindbox secretKey="'+config.mindbox\_api\_key+'"'  
},  
body: '<operation>\r\n <page>\r\n <firstmindboxid>' + startId.toString() + '</firstmindboxid>\r\n <pagenumber>1</pagenumber>\r\n <itemsperpage>1000</itemsperpage>\r\n </page>\r\n</operation>'  
};  
  
request(options, function (error, response, body) {  
cb(error, body);  
});  
}  
  
/\*\*  
\* Обработка блока данных от Mindbox  
\* Записать в таблицу table MsSQL  
\*/  
function mbxHandle(table, rows, cb) {

console.log(rows.length.toString() + ' items to save into '+table);  
var i = 0;  
  
async.forEach(rows, function(row, cb1) {  
try {  
var sqlreq = new sql.Request();  
sqlreq.input('mindboxId', [sql.Int](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fsql.Int" \t "_blank), parseInt(row.ids[0].mindboxId[0]));  
sqlreq.input('transactionId', sql.VarChar, (row.ids[0].transactionId ? row.ids[0].transactionId[0] : ''));  
sqlreq.input('systemName', row.actionTemplate[0].systemName[0]);  
sqlreq.input('name', sql.NVarChar, row.actionTemplate[0].name[0]);  
sqlreq.input('dateTimeUtc', sql.DateTime, new Date(row.dateTimeUtc[0]));  
sqlreq.input('pointOfContact\_externalId', sql.VarChar, row.pointOfContact[0].ids[0].externalId[0]);  
sqlreq.input('customer\_mindboxId', sql.VarChar, (row.customer[0].ids[0] ? row.customer[0].ids[0].mindboxId[0] : ''));  
sqlreq.input('customer\_webSiteId', sql.VarChar, (row.customer[0].ids[0].webSiteId ? row.customer[0].ids[0].webSiteId[0] : ''));  
sqlreq.input('channel', sql.VarChar, (row.mailing && row.mailing[0].channel ? row.mailing[0].channel[0] : ''));  
sqlreq.input('action', sql.VarChar, (row.mailing && row.mailing[0].action ? row.mailing[0].action[0] : ''));  
sqlreq.input('notSentReason\_name', sql.VarChar, row.ids[0].mindboxId[0]);  
sqlreq.input('notSentReason\_systemName', sql.VarChar, (row.mailing && row.mailing[0].notSentReason && row.mailing[0].notSentReason[0].name ? row.mailing[0].notSentReason[0].name[0] : ''));  
sqlreq.input('groupingKey', sql.VarChar, (row.mailing && row.mailing[0].groupingKey ? row.mailing[0].groupingKey[0] : ''));  
sqlreq.input('link', sql.VarChar, (row.mailing && row.mailing[0].link ? row.mailing[0].link[0] : ''));  
sqlreq.input('viewEmailMessageUrl', sql.VarChar, (row.mailing && row.mailing[0].viewEmailMessageUrl ? row.mailing[0].viewEmailMessageUrl[0] : ''));  
var q = "INSERT INTO "+config.mssql.database+".dbo."+table  
+ " (mindboxId, transactionId, systemName, name, dateTimeUtc, pointOfContact\_externalId, customer\_mindboxId, customer\_webSiteId, channel, [action], notSentReason\_name, notSentReason\_systemName, groupingKey, link, viewEmailMessageUrl)"  
+ " VALUES(@mindboxId, @transactionId, @systemName, [@name](https://vk.com/id44899), @dateTimeUtc, @pointOfContact\_externalId, @customer\_mindboxId, @customer\_webSiteId, [@channel](https://vk.com/id3696360), [@action](https://vk.com/id391046), @notSentReason\_name, @notSentReason\_systemName, @groupingKey, @link, @viewEmailMessageUrl);";  
}  
catch (err) {  
console.log('MsSQL input vars error: ', err, JSON.stringify(row));  
return cb1(err);  
}  
sqlreq.query(q, function(err, r) {  
if (err) {  
console.log('MsSQL query error: ', err, JSON.stringify(row));  
}  
cb1(err, r);  
})  
}, function(err, r) {  
cb(err, r);  
});  
}



# **Вывод**

Я изучила представленную тему и разобралась в работе в сфере BI.