- Query Bulider
  - 获得数据
    - get
    - pluck("title","name") 提取一列的值
      - 第二个参数是指定的key
    - 分批处理 chunk

```
DB::table('users')->orderBy('id')->chunk(100, function ($users) {
    foreach ($users as $user) {
        //
//return false 将会停止
    }
});
一定要有orderBy
```

    - value
      - 直接取得值
    - aggregate
      - count, max, min, avg, and sum
    - select

```
DB::table('users')->select('name', 'email as user_email')->get();
```

```
$users = DB::table('users')->distinct()->get();
```

```
$query = DB::table('users')->select('name');
$users = $query->addSelect('age')->get();
```

    - raw expression

```
使用DB::raw 进行算选计算
$users = DB::table('users')
                ->select(DB::raw('count(*) as user_count, status'))
                ->where('status', '<>', 1)
                ->groupBy('status')
                ->get();
```

    - join
      - left join(左联接) 返回包括左表中的所有记录和右表中联结字段相等的记录
        right join(右联接) 返回包括右表中的所有记录和左表中联结字段相等的记录
        inner join(等值连接) 只返回两个表中联结字段相等的行

```
$users = DB::table('users')
            ->join('contacts', 'users.id', '=', 'contacts.user_id')
            ->join('orders', 'users.id', '=', 'orders.user_id')
            ->select('users.*', 'contacts.phone', 'orders.price')
            ->get();
```

```
DB::table('users')
        ->join('contacts', function ($join) {
            $join->on('users.id', '=', 'contacts.user_id')->orOn(...);
        })
        ->get();
```

```
#加一些筛选条件
DB::table('users')
        ->join('contacts', function ($join) {
            $join->on('users.id', '=', 'contacts.user_id')
                 ->where('contacts.user_id', '>', 5);
        })
        ->get();
```

    - where
      - where('votes', '>=', 100)
        - = like > < 等

```
# and
$users = DB::table('users')->where([
    ['status', '=', '1'],
    ['subscribed', '<>', '1'],
])->get();
```

```
# or
$users = DB::table('users')
            ->where('votes', '>', 100)
            ->orWhere('name', 'John')
            ->get();
```

```
$users = DB::table('users')
            ->whereBetween('votes', [1, 100])->get();
```

```
$users = DB::table('users')
            ->whereNotBetween('votes', [1, 100])
            ->get();
```

```
$users = DB::table('users')
            ->whereIn('id', [1, 2, 3])
            ->get();
```

```
$users = DB::table('users')
            ->whereNotIn('id', [1, 2, 3])
            ->get();
```

      - whereNull，whereNotNull，whereDate，whereMonth，whereDay，whereYear，whereColumn，whereNotIn，whereIn，where，orWhere，whereBetween，whereNoBetween，whereExists，whereRaw，
        - may be used to verify that two columns are equal

```
$users = DB::table('users')
            ->whereColumn([
                ['first_name', '=', 'last_name'],
                ['updated_at', '>', 'created_at']
            ])->get();
```

```
DB::table('users')
        ->where('name', '=', 'John')
        ->orWhere(function ($query) {
            $query->where('votes', '>', 100)
                  ->where('title', '<>', 'Admin');
        })
        ->get();
```

        - select * from users where name = 'John' or (votes > 100 and title <> 'Admin')

```
DB::table('users')
        ->whereExists(function ($query) {
            $query->select(DB::raw(1))
                  ->from('orders')
                  ->whereRaw('orders.user_id = users.id');
        })
        ->get();
```

        - select * from `users` where exists(select ``*)
    - order
      - orderBy('name',"desc")
      - latest/ordest　按照created_at 进行排序
      - inRandomOrder
      - groupBy/having/havingRaw

```
$users = DB::table('users')
            ->groupBy('account_id')
            ->having('account_id', '>', 100)
            ->get();
```

```
$users = DB::table('orders')
            ->select('department', DB::raw('SUM(price) as total_sales'))
            ->groupBy('department')
            ->havingRaw('SUM(price) > 2500')
            ->get();
```

      - skip/take
        - 或者limit offset

```
$users = DB::table('users')
            ->offset(10)
            ->limit(5)
            ->get();
```

        - $users = DB::table('users')->skip(10)->take(5)->get();
    - when 条件从句
      - 只有满足条件才会去执行 从句里面的sql

```
$role = $request->input('role');

$users = DB::table('users')
                ->when($role, function ($query) use ($role) {
                    return $query->where('role_id', $role);
                })
                ->get();
```

```
不是执行这个就是执行另外一个
$sortBy = null;

$users = DB::table('users')
                ->when($sortBy, function ($query) use ($sortBy) {
                    return $query->orderBy($sortBy);
                }, function ($query) {
                    return $query->orderBy('name');
                })
                ->get();
```

    - insert

```
DB::table('users')->insert(
    ['email' => 'john@example.com', 'votes' => 0]
);
```

```
DB::table('users')->insert([
    ['email' => 'taylor@example.com', 'votes' => 0],
    ['email' => 'dayle@example.com', 'votes' => 0]
]);
```

```
#insertGetId 获得插入的id 如果有increment_id
$id = DB::table('users')->insertGetId(
    ['email' => 'john@example.com', 'votes' => 0]
);
```

    - update

```
DB::table('users')
            ->where('id', 1)
            -
>update(['votes' => 1]);
```

```
更新JSON 数据类型
DB::table('users')
            ->where('id', 1)
            ->update(['options-
>enabled' => true]);
```

      - increment/decrement 方法
        - 自动加1 或者加具体的数

```
DB::table('users')->increment('votes'); 第二个参数自动是1
DB::table('users')->increment('votes', 5);
DB::table('users')->increment('votes', 1, ['name' => 'John']); 另外的更新
```

    - delete

```
DB::table('users')->delete();
```

```
DB::table('users')->where('votes', '>', 100)->delete();
```

```
DB::table('users')->truncate();
```

    - 锁

```
DB::table('users')->where('votes', '>', 100)->sharedLock()->get();
```

```
DB::table('users')->where('votes', '>', 100)->lockForUpdate()->get();
```