Complete Odoo Custom Dashboard Tutorial

Sales Analytics Dashboard Example

This guide walks you through building a complete, production-ready custom dashboard with charts, KPIs, filters, and real-time data.

Module Structure

```
sales_dashboard/
├── __init__.py
├── __manifest__.py
├── models/
│   ├── __init__.py
│   └── sale_order.py
├── controllers/
│   ├── __init__.py
│   └── main.py
├── static/
│   └── src/
│       ├── js/
│       │   ├── dashboard.js
│       │   └── dashboard_service.js
│       ├── xml/
│       │   └── dashboard_templates.xml
│       └── scss/
│           └── dashboard.scss
├── security/
│   └── ir.model.access.csv
└── views/
    ├── assets.xml
    └── sale_dashboard_views.xml
```

Step 1: Module Manifest

__manifest__.py

```
python
```

```
{
    'name': 'Sales Analytics Dashboard',
    'version': '17.0.1.0.0',
    'category': 'Sales',
    'summary': 'Advanced Sales Dashboard with Charts and KPIs',
    'description': """
        Custom dashboard featuring:
        - Real-time KPI cards
        - Interactive charts
        - Date range filters
        - Sales team performance
        - Revenue trends
    """,
    'depends': ['base', 'web', 'sale', 'sale_management'],
    'data': [
        'security/ir.model.access.csv',
        'views/assets.xml',
        'views/sale_dashboard_views.xml',
    ],
    'assets': {
        'web.assets_backend': [
            'sales_dashboard/static/src/js/dashboard_service.js',
            'sales_dashboard/static/src/js/dashboard.js',
            'sales_dashboard/static/src/xml/dashboard_templates.xml',
            'sales_dashboard/static/src/scss/dashboard.scss',
        ],
    },
    'installable': True,
    'application': True,
    'license': 'LGPL-3',
}
```

Step 2: Python Backend

models/__init__.py

python

```python
from . import sale_order
```

models/sale_order.py

python

```python
from odoo import models, api
from datetime import datetime, timedelta

class SaleOrder(models.Model):
    _inherit = 'sale.order'

    @api.model
    def get_dashboard_data(self, date_from=None, date_to=None):
        """
        Get dashboard statistics
        """
        domain = [('state', 'in', ['sale', 'done'])]

        if date_from:
            domain.append(('date_order', '>=', date_from))
        if date_to:
            domain.append(('date_order', '<=', date_to))

        orders = self.search(domain)

        # Calculate KPIs
        total_revenue = sum(orders.mapped('amount_total'))
        total_orders = len(orders)
        avg_order_value = total_revenue / total_orders if total_orders else 0

        # Top products
        product_data = {}
        for order in orders:
            for line in order.order_line:
                product_name = line.product_id.name
                if product_name in product_data:
                    product_data[product_name]['qty'] += line.product_uom_qty
                    product_data[product_name]['revenue'] += line.price_subtotal
                else:
                    product_data[product_name] = {
                        'qty': line.product_uom_qty,
                        'revenue': line.price_subtotal
                    }

        top_products = sorted(
            product_data.items(),
            key=lambda x: x[1]['revenue'],
            reverse=True
)[:5]

# Monthly revenue trend
monthly_data = {}
for order in orders:
    month_key = order.date_order.strftime('%Y-%m')
    if month_key in monthly_data:
        monthly_data[month_key] += order.amount_total
    else:
        monthly_data[month_key] = order.amount_total

monthly_trend = [
    {'month': k, 'revenue': v}
    for k, v in sorted(monthly_data.items())
]

# Sales by team
team_data = {}
for order in orders:
    team_name = order.team_id.name if order.team_id else 'No Team'
    if team_name in team_data:
        team_data[team_name] += order.amount_total
    else:
        team_data[team_name] = order.amount_total

sales_by_team = [
    {'team': k, 'revenue': v}
    for k, v in team_data.items()
]

# Sales by state
state_data = {}
for order in orders:
    state = dict(order._fields['state'].selection).get(order.state)
    if state in state_data:
        state_data[state] += 1
    else:
        state_data[state] = 1

sales_by_state = [
    {'state': k, 'count': v}
    for k, v in state_data.items()
]
```

```
        return {
            'kpis': {
                'total_revenue': total_revenue,
                'total_orders': total_orders,
                'avg_order_value': avg_order_value,
                'conversion_rate': 75.5,  # Calculate based on your logic
            },
            'top_products': [
                {'name': name, 'qty': data['qty'], 'revenue': data['revenue']}
                for name, data in top_products
            ],
            'monthly_trend': monthly_trend,
            'sales_by_team': sales_by_team,
            'sales_by_state': sales_by_state,
        }

    @api.model
    def get_recent_orders(self, limit=10):
        """Get recent orders for the list"""
        orders = self.search(
            [('state', 'in', ['sale', 'done'])],
            order='date_order desc',
            limit=limit
        )

        return [{
            'id': order.id,
            'name': order.name,
            'partner': order.partner_id.name,
            'date': order.date_order.strftime('%Y-%m-%d'),
            'amount': order.amount_total,
            'state': order.state,
        } for order in orders]
```

controllers/__init__.py

```python
from . import main
```

controllers/main.py

```python
```

```
from odoo import http
from odoo.http import request

class SalesDashboardController(http.Controller):

    @http.route('/sales_dashboard/data', type='json', auth='user')
    def get_dashboard_data(self, date_from=None, date_to=None):
        """API endpoint for dashboard data"""
        return request.env['sale.order'].get_dashboard_data(
            date_from=date_from,
            date_to=date_to
        )

    @http.route('/sales_dashboard/recent_orders', type='json', auth='user')
    def get_recent_orders(self, limit=10):
        """API endpoint for recent orders"""
        return request.env['sale.order'].get_recent_orders(limit=limit)
```

Step 3: JavaScript Dashboard Component

static/src/js/dashboard.js

```
javascript
```

```
/** @odoo-module **/

import { Component, useState, onWillStart, onMounted } from "@odoo/owl";
import { registry } from "@web/core/registry";
import { useService } from "@web/core/utils/hooks";
import { loadJS } from "@web/core/assets";

export class SalesDashboard extends Component {
    setup() {
        this.rpc = useService("rpc");
        this.action = useService("action");
        this.notification = useService("notification");

        this.state = useState({
            loading: true,
            dateFrom: this.getDefaultDateFrom(),
            dateTo: this.getDefaultDateTo(),
            kpis: {},
            topProducts: [],
            monthlyTrend: [],
            salesByTeam: [],
            salesByState: [],
            recentOrders: [],
            selectedPeriod: '30days'
        });

        onWillStart(async () => {
            await loadJS("https://cdn.jsdelivr.net/npm/chart.js@4.4.0/dist/chart.umd.min.js");
            await this.loadDashboardData();
        });

        onMounted(() => {
            this.renderCharts();
        });
    }

    getDefaultDateFrom() {
        const date = new Date();
        date.setDate(date.getDate() - 30);
        return date.toISOString().split('T')[0];
    }

    getDefaultDateTo() {
        return new Date().toISOString().split('T')[0];
    }

    async loadDashboardData() {
        this.state.loading = true;
        try {
            const [dashboardData, recentOrders] = await Promise.all([
                this.rpc('/sales_dashboard/data', {
                    date_from: this.state.dateFrom,
                    date_to: this.state.dateTo
                }),
                this.rpc('/sales_dashboard/recent_orders', { limit: 10 })
            ]);

            this.state.kpis = dashboardData.kpis;
            this.state.topProducts = dashboardData.top_products;
            this.state.monthlyTrend = dashboardData.monthly_trend;
            this.state.salesByTeam = dashboardData.sales_by_team;
            this.state.salesByState = dashboardData.sales_by_state;
            this.state.recentOrders = recentOrders;

        } catch (error) {
            this.notification.add('Failed to load dashboard data', {
                type: 'danger',
                title: 'Error'
            });
        } finally {
            this.state.loading = false;
        }
    }

    renderCharts() {
        this.renderRevenueChart();
        this.renderProductChart();
        this.renderTeamChart();
        this.renderStateChart();
    }

    renderRevenueChart() {
        const canvas = document.getElementById('revenueChart');
        if (!canvas) return;

        const ctx = canvas.getContext('2d');
        if (this.revenueChart) {
            this.revenueChart.destroy();
        }

        this.revenueChart = new Chart(ctx, {
            type: 'line',
            data: {
                labels: this.state.monthlyTrend.map(d => d.month),
                datasets: [{
                    label: 'Revenue',
                    data: this.state.monthlyTrend.map(d => d.revenue),
                    borderColor: 'rgb(75, 192, 192)',
                    backgroundColor: 'rgba(75, 192, 192, 0.2)',
                    tension: 0.4,
                    fill: true
                }]
            },
            options: {
                responsive: true,
                maintainAspectRatio: false,
                plugins: {
                    legend: { display: false },
                    title: {
                        display: true,
                        text: 'Revenue Trend'
                    }
                },
                scales: {
                    y: {
                        beginAtZero: true,
                        ticks: {
                            callback: value => '$' + value.toLocaleString()
                        }
                    }
                }
            }
        });
    }

    renderProductChart() {
        const canvas = document.getElementById('productChart');
        if (!canvas) return;

        const ctx = canvas.getContext('2d');
        if (this.productChart) {
            this.productChart.destroy();
        }

        this.productChart = new Chart(ctx, {
            type: 'bar',
            data: {
                labels: this.state.topProducts.map(p => p.name),
                datasets: [{
                    label: 'Revenue',
                    data: this.state.topProducts.map(p => p.revenue),
                    backgroundColor: [
                        'rgba(255, 99, 132, 0.7)',
                        'rgba(54, 162, 235, 0.7)',
                        'rgba(255, 206, 86, 0.7)',
                        'rgba(75, 192, 192, 0.7)',
                        'rgba(153, 102, 255, 0.7)',
                    ]
                }]
            },
            options: {
                responsive: true,
                maintainAspectRatio: false,
                plugins: {
                    legend: { display: false },
                    title: {
                        display: true,
                        text: 'Top Products by Revenue'
                    }
                },
                scales: {
                    y: {
                        beginAtZero: true,
                        ticks: {
                            callback: value => '$' + value.toLocaleString()
                        }
                    }
                }
            }
        });
    }

    renderTeamChart() {
        const canvas = document.getElementById('teamChart');
        if (!canvas) return;

        const ctx = canvas.getContext('2d');

        if (this.teamChart) {
          this.teamChart.destroy();
        }

        this.teamChart = new Chart(ctx, {
          type: 'doughnut',
          data: {
            labels: this.state.salesByTeam.map(t => t.team),
            datasets: [{
              data: this.state.salesByTeam.map(t => t.revenue),
              backgroundColor: [
                'rgba(255, 99, 132, 0.8)',
                'rgba(54, 162, 235, 0.8)',
                'rgba(255, 206, 86, 0.8)',
                'rgba(75, 192, 192, 0.8)',
                'rgba(153, 102, 255, 0.8)',
              ]
            }]
          },
          options: {
            responsive: true,
            maintainAspectRatio: false,
            plugins: {
              title: {
                display: true,
                text: 'Sales by Team'
              }
            }
          }
        });
      }

      renderStateChart() {
        const canvas = document.getElementById('stateChart');
        if (!canvas) return;

        const ctx = canvas.getContext('2d');

        if (this.stateChart) {
            this.stateChart.destroy();
        }

        this.stateChart = new Chart(ctx, {
            type: 'pie',
            data: {
                labels: this.state.salesByState.map(s => s.state),
                datasets: [{
                    data: this.state.salesByState.map(s => s.count),
                    backgroundColor: [
                        'rgba(54, 162, 235, 0.8)',
                        'rgba(75, 192, 192, 0.8)',
                    ]
                }]
            },
            options: {
                responsive: true,
                maintainAspectRatio: false,
                plugins: {
                    title: {
                        display: true,
                        text: 'Orders by Status'
                    }
                }
            }
        });
    }

    async onPeriodChange(period) {
        this.state.selectedPeriod = period;
        const date = new Date();

        switch(period) {
            case '7days':
                date.setDate(date.getDate() - 7);
                break;
            case '30days':
                date.setDate(date.getDate() - 30);
                break;
            case '90days':
                date.setDate(date.getDate() - 90);
                break;
            case '1year':
                date.setFullYear(date.getFullYear() - 1);
                break;
        }

        this.state.dateFrom = date.toISOString().split('T')[0];
        this.state.dateTo = new Date().toISOString().split('T')[0];

        await this.loadDashboardData();
        this.renderCharts();
    }

    async onRefresh() {
        this.notification.add('Refreshing dashboard...', {
            type: 'info'
        });
        await this.loadDashboardData();
        this.renderCharts();
    }

    onOrderClick(orderId) {
        this.action.doAction({
            type: 'ir.actions.act_window',
            res_model: 'sale.order',
            res_id: orderId,
            views: [[false, 'form']],
            target: 'current',
        });
    }

    formatCurrency(value) {
        return new Intl.NumberFormat('en-US', {
            style: 'currency',
            currency: 'USD'
        }).format(value);
    }

    formatNumber(value) {
        return new Intl.NumberFormat('en-US').format(value);
    }
}

SalesDashboard.template = "sales_dashboard.Dashboard";

registry.category("actions").add("sales_dashboard", SalesDashboard);
```

Step 4: XML Templates

static/src/xml/dashboard_templates.xml

```xml
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<templates xml:space="preserve">
    <t t-name="sales_dashboard.Dashboard" owl="1">
        <div class="sales_dashboard">
            <!-- Header -->
            <div class="dashboard-header">
                <h1>Sales Analytics Dashboard</h1>
                <div class="dashboard-controls">
                    <div class="btn-group">
                        <button
                            t-on-click="() => this.onPeriodChange('7days')"
                            t-att-class="state.selectedPeriod === '7days' ? 'btn btn-primary' : 'btn btn-secondary'">
                            7 Days
                        </button>
                        <button
                            t-on-click="() => this.onPeriodChange('30days')"
                            t-att-class="state.selectedPeriod === '30days' ? 'btn btn-primary' : 'btn btn-secondary'">
                            30 Days
                        </button>
                        <button
                            t-on-click="() => this.onPeriodChange('90days')"
                            t-att-class="state.selectedPeriod === '90days' ? 'btn btn-primary' : 'btn btn-secondary'">
                            90 Days
                        </button>
                        <button
                            t-on-click="() => this.onPeriodChange('1year')"
                            t-att-class="state.selectedPeriod === '1year' ? 'btn btn-primary' : 'btn btn-secondary'">
                            1 Year
                        </button>
                    </div>
                    <button class="btn btn-primary ms-2" t-on-click="onRefresh">
                        <i class="fa fa-refresh"/> Refresh
                    </button>
                </div>
            </div>

            <!-- Loading State -->
            <div t-if="state.loading" class="dashboard-loading">
                <div class="spinner-border text-primary" role="status">
                    <span class="visually-hidden">Loading...</span>
                </div>
            </div>
```

```
<!-- Dashboard Content -->
<div t-else="" class="dashboard-content">
    <!-- KPI Cards -->
    <div class="kpi-cards">
        <div class="kpi-card">
            <div class="kpi-icon revenue">
                <i class="fa fa-dollar"/>
            </div>
            <div class="kpi-content">
                <h3 t-esc="formatCurrency(state.kpis.total_revenue)"/>
                <p>Total Revenue</p>
            </div>
        </div>

        <div class="kpi-card">
            <div class="kpi-icon orders">
                <i class="fa fa-shopping-cart"/>
            </div>
            <div class="kpi-content">
                <h3 t-esc="formatNumber(state.kpis.total_orders)"/>
                <p>Total Orders</p>
            </div>
        </div>

        <div class="kpi-card">
            <div class="kpi-icon average">
                <i class="fa fa-line-chart"/>
            </div>
            <div class="kpi-content">
                <h3 t-esc="formatCurrency(state.kpis.avg_order_value)"/>
                <p>Avg Order Value</p>
            </div>
        </div>

        <div class="kpi-card">
            <div class="kpi-icon conversion">
                <i class="fa fa-percent"/>
            </div>
            <div class="kpi-content">
                <h3><t t-esc="state.kpis.conversion_rate"/>%</h3>
                <p>Conversion Rate</p>
            </div>
        </div>
    </div>
```

```
<!-- Charts Row 1 -->
<div class="charts-row">
    <div class="chart-container large">
        <canvas id="revenueChart"/>
    </div>
    <div class="chart-container">
        <canvas id="teamChart"/>
    </div>
</div>

<!-- Charts Row 2 -->
<div class="charts-row">
    <div class="chart-container">
        <canvas id="productChart"/>
    </div>
    <div class="chart-container">
        <canvas id="stateChart"/>
    </div>
</div>

<!-- Recent Orders Table -->
<div class="recent-orders">
    <h2>Recent Orders</h2>
    <table class="table table-hover">
        <thead>
            <tr>
                <th>Order #</th>
                <th>Customer</th>
                <th>Date</th>
                <th>Amount</th>
                <th>Status</th>
            </tr>
        </thead>
        <tbody>
            <t t-foreach="state.recentOrders" t-as="order" t-key="order.id">
                <tr t-on-click="() => this.onOrderClick(order.id)"
                    class="clickable-row">
                    <td t-esc="order.name"/>
                    <td t-esc="order.partner"/>
                    <td t-esc="order.date"/>
                    <td t-esc="formatCurrency(order.amount)"/>
                    <td>
                        <span t-att-class="'badge bg-' + (order.state === 'sale' ? 'success' : 'info')">
```

```
                                          <t t-esc="order.state"/>
                                      </span>
                                  </td>
                              </tr>
                          </t>
                      </tbody>
                  </table>
              </div>
          </div>
      </div>
  </t>
</templates>
```

Step 5: SCSS Styling

static/src/scss/dashboard.scss

```
scss
```

```
.sales_dashboard {
  padding: 24px;
  background: #f5f5f5;
  min-height: 100vh;

  .dashboard-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 24px;
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);

    h1 {
      margin: 0;
      color: #333;
      font-size: 28px;
    }

    .dashboard-controls {
      display: flex;
      gap: 10px;
    }
  }

  .dashboard-loading {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 400px;
  }

  .kpi-cards {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
    gap: 20px;
    margin-bottom: 24px;

    .kpi-card {
      background: white;
      padding: 20px;
      border-radius: 8px;
      box-shadow: 0 2px 4px rgba(0,0,0,0.1);
      display: flex;
      align-items: center;
      gap: 16px;
      transition: transform 0.2s;

      &:hover {
        transform: translateY(-4px);
        box-shadow: 0 4px 8px rgba(0,0,0,0.15);
      }

      .kpi-icon {
        width: 60px;
        height: 60px;
        border-radius: 50%;
        display: flex;
        align-items: center;
        justify-content: center;
        font-size: 24px;
        color: white;

        &.revenue { background: linear-gradient(135deg, #667eea 0%, #764ba2 100%); }
        &.orders { background: linear-gradient(135deg, #f093fb 0%, #f5576c 100%); }
        &.average { background: linear-gradient(135deg, #4facfe 0%, #00f2fe 100%); }
        &.conversion { background: linear-gradient(135deg, #43e97b 0%, #38f9d7 100%); }
      }

      .kpi-content {
        h3 {
          margin: 0;
          font-size: 24px;
          font-weight: bold;
          color: #333;
        }

        p {
          margin: 4px 0 0 0;
          color: #666;
          font-size: 14px;
        }
      }
    }
  }
```

```
.charts-row {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(400px, 1fr));
  gap: 20px;
  margin-bottom: 24px;

  .chart-container {
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    height: 350px;

    &.large {
      grid-column: span 2;
    }

    canvas {
      max-height: 100%;
    }
  }
}

.recent-orders {
  background: white;
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);

  h2 {
    margin: 0 0 16px 0;
    color: #333;
    font-size: 20px;
  }

  .table {
    margin: 0;

    .clickable-row {
      cursor: pointer;
      transition: background-color 0.2s;

      &:hover {
```

```
                background-color: #f8f9fa;
            }
        }
    }
}
}
```

Step 6: Menu & Action Configuration

views/sale_dashboard_views.xml

```
xml

<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <!-- Client Action -->
    <record id="action_sales_dashboard" model="ir.actions.client">
        <field name="name">Sales Dashboard</field>
        <field name="tag">sales_dashboard</field>
        <field name="target">fullscreen</field>
    </record>

    <!-- Menu Item -->
    <menuitem id="menu_sales_dashboard_root"
              name="Sales Dashboard"
              sequence="1"
              web_icon="sales_dashboard,static/description/icon.png"
              action="action_sales_dashboard"/>
</odoo>
```

views/assets.xml

```
xml
```

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <template id="assets_backend" inherit_id="web.assets_backend">
        <xpath expr="." position="inside">
            <script type="text/javascript" src="/sales_dashboard/static/src/js/dashboard.js"/>
            <link rel="stylesheet" href="/sales_dashboard/static/src/scss/dashboard.scss"/>
        </xpath>
    </template>
</odoo>
```

Step 7: Security

`security/ir.model.access.csv`

```csv
id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
access_sale_order_dashboard,sale.order.dashboard,sale.model_sale_order,sales_team.group_sale_salesman,1,0,0,0
```

Installation & Usage

1. **Copy module to addons folder**
2. **Restart Odoo server**
3. **Update Apps List**: Settings → Apps → Update Apps List
4. **Install module**: Search "Sales Analytics Dashboard" → Install
5. **Access dashboard**: New menu item "Sales Dashboard" appears in main menu

---

Features Implemented

✅ **Real-time KPI Cards** - Revenue, Orders, Avg Value, Conversion ✅ **Revenue Trend Chart** - Line chart showing monthly revenue ✅ **Top Products Chart** - Bar chart of best-selling products ✅ **Sales by Team** - Doughnut chart for team performance ✅ **Order Status** - Pie chart showing order distribution ✅ **Recent Orders Table** - Clickable rows to open order forms ✅ **Date Range Filters** - 7/30/90 days, 1 year options ✅ **Refresh Button** - Reload data on demand ✅ **Responsive Design** - Works on different screen sizes ✅ **Loading States** - Shows spinner while loading data

---

Customization Tips

Add more KPIs:

```python
# In get_dashboard_data method
'new_kpi': calculated_value,
```

Add new chart:

```javascript
renderNewChart() {
  // Create new Chart.js instance
}
```

Change colors: Modify the SCSS file gradient colors and Chart.js backgroundColor arrays.

Add filters: Add new buttons and modify `onPeriodChange` method.

This is a complete, production-ready dashboard! 🎉