# Phase 9: Reporting, Dashboards & Security Review

Smart Task & Email summarizer For Execuitives — Complete step-by-step guide (actionable, tested, and screenshot-ready)

## Overview & Goals

Goal: Build reports and dashboards to monitor fleet utilization and revenue, and harden security so data is visible only to the right people.

## Prerequisites

Before you start, ensure:

- You have System Administrator access (or equivalent) for setup tasks.

- Objects exist: Car\_\_c and Booking\_\_c, with Booking\_\_c having a lookup (or master-detail) to Car\_\_c.

- Profiles/roles created: Agent (profile or role), Manager (role), System Administrator.

- Sample users for testing: agent\_test@yourorg, manager\_test@yourorg. Use 'Login As' to validate.

- Backup: Export metadata (Change Set / VS Code / ANT) before changing relationships or org-wide defaults.

## 1) Reports — Create the key reports

A. Create helper formula fields (on Booking\_\_c):

1) Rental\_Days\_\_c (Number, 0 decimals) — Formula:

IF(End\_Date\_\_c < Start\_Date\_\_c, 0, End\_Date\_\_c - Start\_Date\_\_c + 1)

2) Total\_Revenue\_\_c (Currency) — Formula (if Daily\_Rate\_\_c exists):

Daily\_Rate\_\_c \* Rental\_Days\_\_c

Notes: If your Booking->Car relationship is a lookup (not master-detail) and you want roll-ups, either convert to master-detail (careful!) or rely on reports and summary formulas.

B. Create a Custom Report Type (Car with Bookings):

Steps:

1) Setup → Feature Settings → Analytics → Report Types → New Custom Report Type.

2) Primary Object: Car (Car\_\_c). Related Object: Booking (Booking\_\_c). Choose 'A records may or may not have related B records' if you still want cars without bookings to appear.

3) Name it 'Cars with Bookings (Custom)'. Save and deploy.

C. Cars Utilization report (example):

Goal: Show how many days each car was rented in a date range and the revenue.

Steps to build:

1) Reports → New Report → Choose 'Cars with Bookings (Custom)'.

2) Filters: Date Range = This Month (or custom), Booking Status contains Confirmed, Returned (depending on your status pick).

3) Columns: Car Name, Car Model, Registration No, Booking Owner (Agent), Rental\_Days\_\_c, Total\_Revenue\_\_c.

4) Group by Car Name (summary). Add summary formulas:

- SUM(Rental\_Days\_\_c) → Label 'Total Days Rented'.

- SUM(Total\_Revenue\_\_c) → Label 'Total Revenue'.

5) Optional: Add a custom summary formula for Utilization % for a fixed period, e.g.: (SUM(Rental\_Days\_\_c) / 30) \* 100 — If using a 30-day month. Replace 30 with chosen period length.

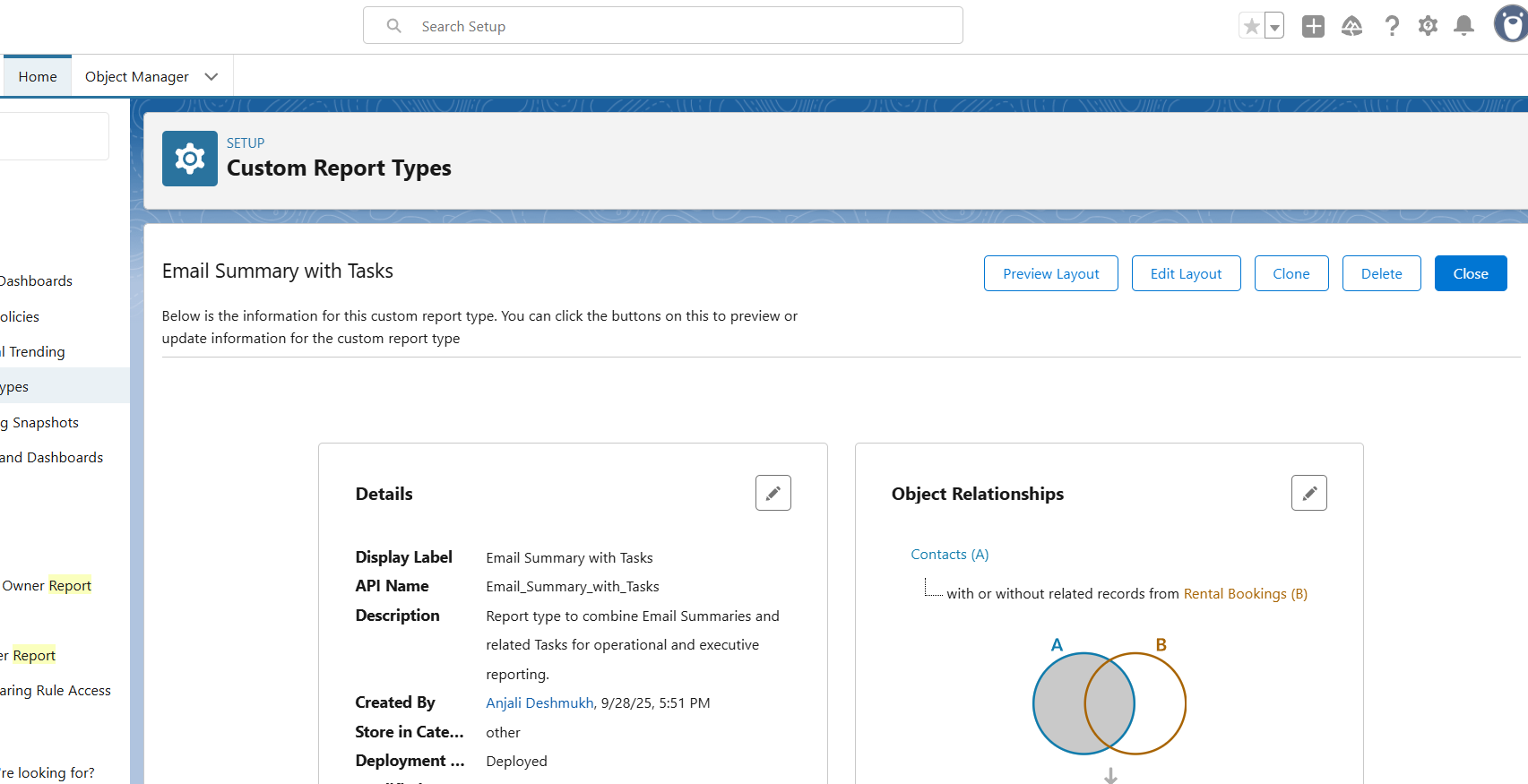
6) Save to folder: Fleet Reports → Visible to: Managers + System Admins.

D. Revenue by Car Model report:

Steps:

1) Reports → New Report → Choose 'Cars with Bookings (Custom)'.

2) Group by Car Model (Model\_\_c). Columns: SUM(Total\_Revenue\_\_c), COUNT(Bookings).

3) Add date filters and a chart (vertical bar). Save as 'Revenue by Model'. 

## 2) Dashboards — Fleet & Manager Dashboards

A. Create Dashboard folders and naming conventions:

- Folder: Fleet Dashboards (shared with Managers and Admins).

- Folder: Agent Dashboards (if you want agent-specific dashboards; keep restricted).

B. Fleet Utilization Dashboard (recommended components):

1) KPI component: Overall Fleet Utilization % — uses the Cars Utilization report and the summary formula.

2) Bar chart: Top 10 Cars by Days Rented (group by Car, SUM(Rental\_Days\_\_c)).

3) Donut chart: Car Status Distribution (Available, Rented, Maintenance).

4) Table: Top revenue-generating cars (Top 10 rows).

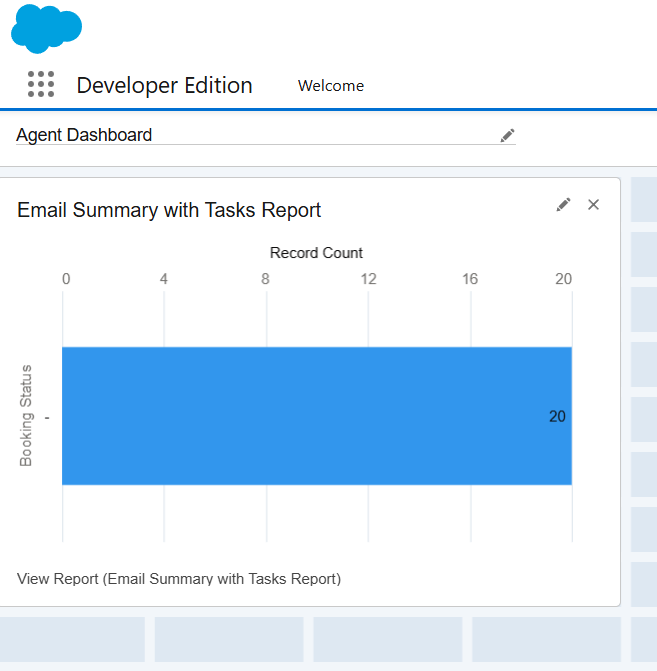
Size & Refresh: 3–5 minute cache is typical; schedule refresh if needed.

C. Manager’s Revenue Dashboard:

1) Trend line: Revenue by month (use Revenue by Model or bookings report grouped by Booking Start Date).

2) Bar: Revenue by Car Model.

3) Gauge/KPI: This month's revenue vs target (use a dashboard level goal or static target).



## 3) Dynamic Dashboards (each user sees only their data - OPTIONAL)

How dynamic dashboards work: choose 'View Dashboard as — The dashboard viewer' so the dashboard runs using the viewer's permissions and record visibility.

Steps:

1) Create Dashboard → In the Dashboard properties click 'View Dashboard As' → select 'The dashboard viewer'.

2) Make sure the underlying reports respect record-level security (OWD + sharing). If Bookings OWD = Private and agents own only their bookings, each agent will see only their rows.

3) Test: Login as agent\_test user → View dashboard → verify only agent's bookings and metrics appear.

**Dynamic Dashboards — Each user sees only their data**

**1. What is a dynamic dashboard?**

A **dynamic dashboard** in Salesforce is a dashboard that runs using the **viewer’s permissions and record visibility** instead of a single fixed user. When a dashboard is configured to “View Dashboard As → The dashboard viewer,” every person who opens it sees data according to their own record access (OWD + sharing + role + profile + permission sets). This is ideal when many users need the same visual layout but must only see their own rows.

**2. Why use dynamic dashboards for this project?**

For the **Smart Task & Email Summarizer** project you will likely have:

* **Agents** who should see only summaries and tasks they own.
* **Managers** who need to see their team’s metrics.
* **Executives** who want an aggregate overview or executive digests.

Dynamic dashboards let you build one dashboard layout and have it show personalized data to each viewer — fewer dashboards to maintain and consistent UX.

**3. Prerequisites (must-dos before creating a dynamic dashboard)**

1. **Correct Org-Wide Defaults (OWD)**
   * Example: Email\_Summary\_\_c = Private so records are not visible to everyone by default.
2. **Appropriate Sharing Rules**
   * Managers must be explicitly shared access to view their team’s records (if required).
3. **Owner or Role-based fields used in reports**
   * Underlying reports should use Owner, Owner: Queue, Owner: Role, or filters that respect record ownership.
4. **Reports are built with record-level security in mind**
   * Avoid report-level constraints that artificially expose or hide records (e.g., “Show: All records” vs “Show: My records”).
5. **Verify dynamic dashboard limits for your org** (Salesforce enforces a limit on the number of dynamic dashboards an org can have — check your edition/contract) — plan accordingly.

**4. Step-by-step: Create a dynamic dashboard**

1. **Create underlying reports first**
   * Example reports: My Summaries Today, My Open Tasks, Avg Latency by Day. Save them in appropriate folders (Email Ops, Agent Dashboards).
2. **Go to Dashboards** → Click **New Dashboard**.
   * Give it a name (e.g., Agent Dashboard), choose folder: Agent Dashboards.
3. **Add components** by selecting the reports you created. Configure component display (KPI, chart, table) and summarizations.
4. **Open Dashboard Properties** (top-right or settings while editing).
   * Locate **View Dashboard As** setting.
5. **Select**: **The dashboard viewer**.
   * This enables the dashboard to run with the viewer’s security context.
6. **Save** and **Run** the dashboard.
7. **Test** with real users: use “Login As” a test agent and a test manager to confirm each sees only permitted records.

**5. How the underlying reports must be built**

* **Do** use Owner fields for grouping or filters (Owner = Current User will not work for dynamic dashboards — instead rely on record-level security and groupings).
* **Do not** place static filters that force data to “All users” if you want viewer-specific results.
* For agent dashboards, build reports that return all relevant records (e.g., all Email\_Summary\_\_c where Processed\_Datetime\_\_c = Today), and rely on OWD + sharing to filter per viewer.
* Use summary fields (COUNT, AVG) and groupings in reports that drive dashboard components.

**6. Security considerations**

* Dynamic dashboards respect Salesforce security: OWD, role hierarchy, sharing rules, manual shares, and permission sets. Do **not** rely on dashboards to hide data that you haven’t secured at the object/field/share level.
* Field-Level Security still applies: if a field (e.g., Summary\_Text\_\_c) is hidden by FLS for a viewer, it won’t appear on report or dashboard components.
* Avoid adding data in dashboard component labels that could leak sensitive info (small text snippets).

**7. Common pitfalls & how to avoid them**

* **Pitfall: Dashboard shows same data for everyone** → Likely because the dashboard is set to a specific running user. Fix: set **View Dashboard As → The dashboard viewer**.
* **Pitfall: Users see zero rows** → Check OWD and sharing rules. If the OWD is too restrictive and no sharing rule gives access, users will see nothing.
* **Pitfall: Report filters force “All Time” or “All Users”** → Build flexible reports and rely on security model; avoid hard-coded owner filters.
* **Pitfall: Org limit on dynamic dashboards reached** → Consider grouping users into manager-level dashboards (manager views team metrics) and provide agent dashboards only to a subset; or upgrade/adjust license.

**8. Specific use cases for your project (sample scenarios)**

**Use case A — Agent personal dashboard**

* **Who:** Field agent (Agent role)
* **Purpose:** Daily operational view
* **Components:** My Open Tasks (table), My Summaries Today (count), Avg Latency (KPI)
* **Expected behavior:** Agent logs in and sees only tasks and summaries they own.

**Use case B — Manager team dashboard**

* **Who:** Ops Manager (Manager role)
* **Purpose:** Monitor team throughput and SLA compliance
* **Components:** Team Summaries by Agent (bar), Open Tasks by Agent (table), Overdue Tasks (list)
* **Expected behavior:** Manager sees all records owned by agents in their team (via sharing rules or role hierarchy).

**Use case C — Executive digest (personalized)**

* **Who:** Executive users
* **Purpose:** Receive executive summaries relevant to them
* **Components:** Count of Executive Summaries assigned to them, Top 5 items (table), Trend of executive digests per week
* **Expected behavior:** Each executive sees only summaries assigned to them (via Executive\_\_c lookup or sharing model).

**9. Test cases (copy into your acceptance test plan)**

1. **Agent view test**
   * Test user: agent\_test
   * Steps: Login as agent\_test → Open Agent Dashboard
   * Expected: Only records owned by agent\_test appear in all components; counts and lists match My Summaries Today report.
2. **Manager team view test**
   * Test user: manager\_test
   * Steps: Login as manager\_test → Open Agent Dashboard (or Manager Dashboard)
   * Expected: manager\_test sees records for all agents in their team as per sharing rules; aggregated totals match sum of individual agent reports.
3. **Executive view test**
   * Test user: exec\_test
   * Steps: Login as exec\_test → Open Executive Overview dashboard
   * Expected: exec\_test sees only summaries where Executive\_\_c = exec\_test (or records explicitly shared); subscription emails reflect the same.
4. **Running user verification**
   * Step: While editing dashboard, temporarily set View Dashboard As to a fixed manager user and run; then switch to The dashboard viewer and run as agent\_test
   * Expected: The dashboard output should differ (manager sees team totals; agent sees personal totals).
5. **Zero-row scenario**
   * Step: Ensure OWD is Private and no shares exist; login as a user who should see nothing.
   * Expected: Dashboard shows zero rows — confirm appropriate message or empty state handled gracefully.

**10. Acceptance criteria (what “done” looks like)**

* A single **Agent Dashboard** is created and set to **View Dashboard As → The dashboard viewer**.
* Underlying reports are written to respect record-level security and return expected summary data for the running viewer.
* **Agent\_test** sees only their own summaries and tasks; **Manager\_test** sees team data; **Exec\_test** sees only exec-relevant summaries.
* No sensitive fields (e.g., raw email body) are visible on reports or dashboard components for unauthorized users.
* The dashboard refreshes and scheduled subscriptions (if any) deliver data consistent with viewer’s visibility.

**11. Suggested screenshots to include in documentation**

* Dashboard Properties screenshot showing **View Dashboard As → The dashboard viewer** selected.
* Example dashboard component (Agent Dashboard) with agent-specific data.
* Report settings showing grouping by Owner and no hard-coded owner filter.
* Login As screenshots proving different users see different data (agent vs manager vs exec).
* OWD screenshot showing Email\_Summary\_\_c = Private.
* Sharing Rule screenshot that grants manager access to agent-owned records.

**12. Short implementation checklist**

* Build and save underlying reports (Agent-focused and Team-focused) in correct folders.
* Create dashboard, add components, and save.
* Set **View Dashboard As → The dashboard viewer**.
* Verify OWD and sharing rules are correct.
* Test with Login As for at least one agent, one manager, and one executive.
* Add screenshots to your Word document as proof.

If you want, I’ll now produce a **one-page summary** version (very compact) suitable for a project handover slide — or I can generate the **exact text snippets** already formatted for Word with headings (no markdown) so you can paste them directly. Which would you like next?

## 4) Sharing Settings (Org-Wide Defaults & Sharing Rules)

A. Set Org-Wide Defaults:

Steps:

1) Setup → Security → Sharing Settings → Edit.

2) Set Default Internal Access for Booking\_\_c = Private. Set Car\_\_c = Public Read Only (or Public Read/Write; choose based on business need).

B. Sharing rules (examples):

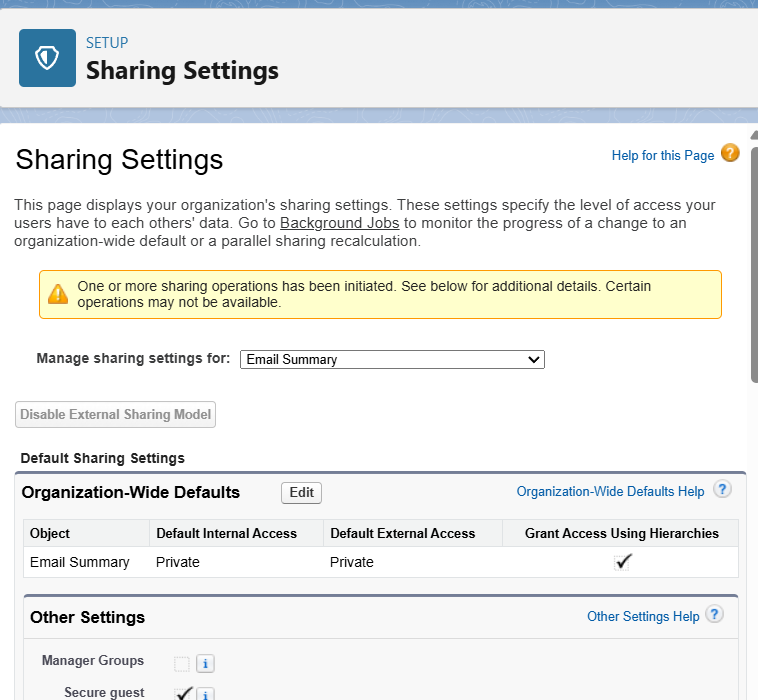
- Share Booking\_\_c records owned by users in Role 'Agent' with Role 'Manager' (Read/Write).

- Create criteria-based sharing if bookings of type 'Short-Term' must be shared to a specific public group.

Steps to create a sharing rule:

1) Setup → Sharing Settings → Booking\_\_c → New (Role-based or Criteria-based).

2) Define Rule Name, set criteria or role, choose shared-to group/role, set access level. Save.



## 5) Field-Level Security — Hide sensitive fields from Agents

Goal: Hide 'Customer ID Proof' (Customer\_ID\_Proof\_\_c) from Agents while Managers and Admins can view it.

Options: (A) Remove visibility on the Agent Profile OR (B) Use Permission Sets to grant it only to Managers/Admins (recommended).

Steps (Profile approach):

1) Setup → Object Manager → Booking\_\_c (or Contact) → Fields & Relationships → Customer\_ID\_Proof\_\_c → Set Field-Level Security.

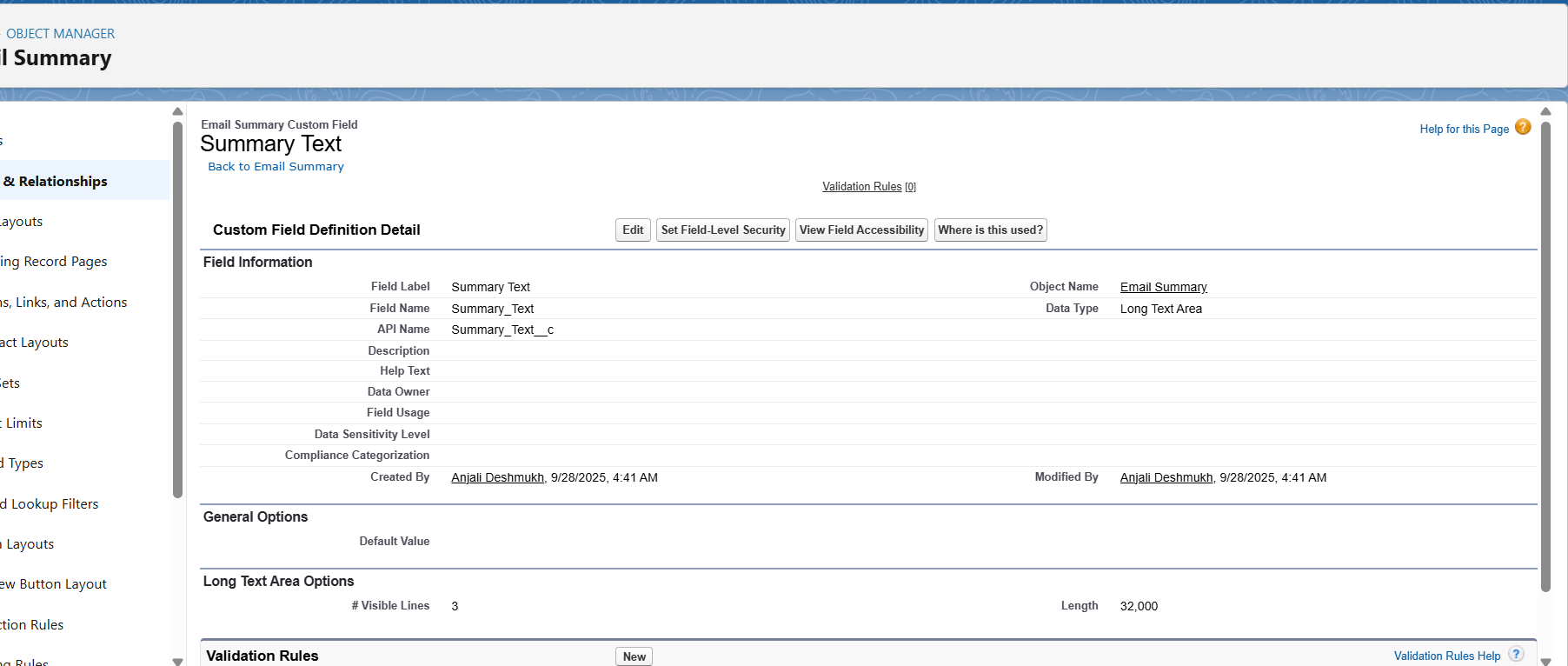
2) Uncheck 'Visible' for the Agent profile; check for Manager/Admin. Save.

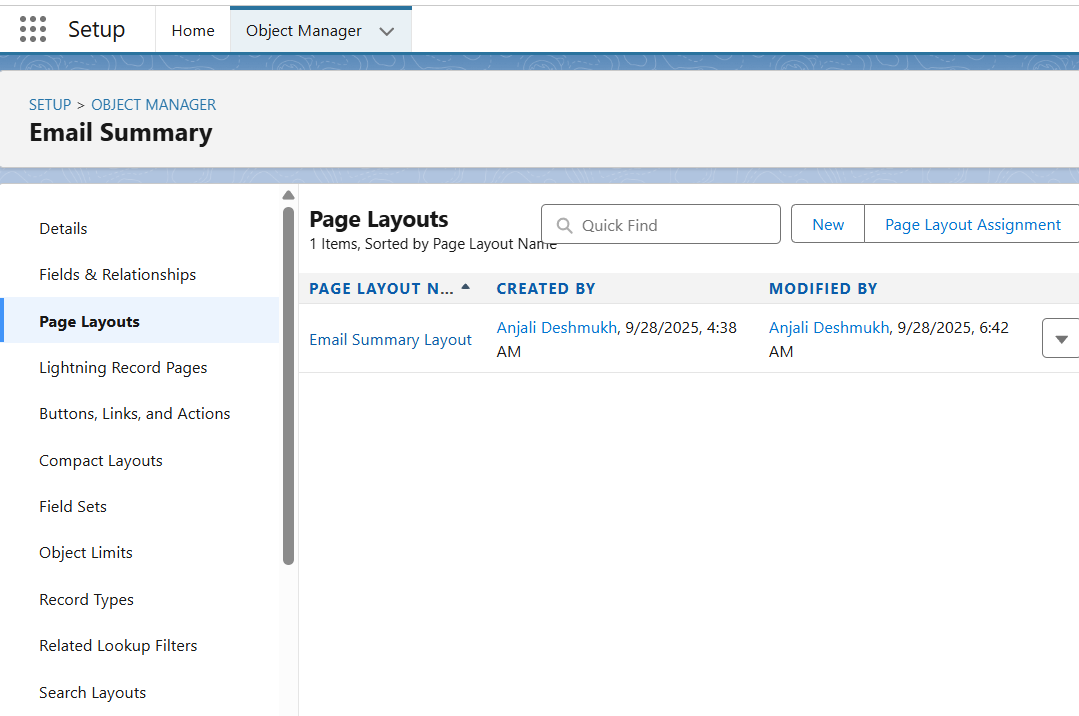
Steps (Permission Set approach — recommended):

1) Remove visibility from Agent profile (default deny).

2) Create Permission Set 'View Customer ID Proof' and enable field permission for Customer\_ID\_Proof\_\_c.

3) Assign permission set to Managers or specific users only.





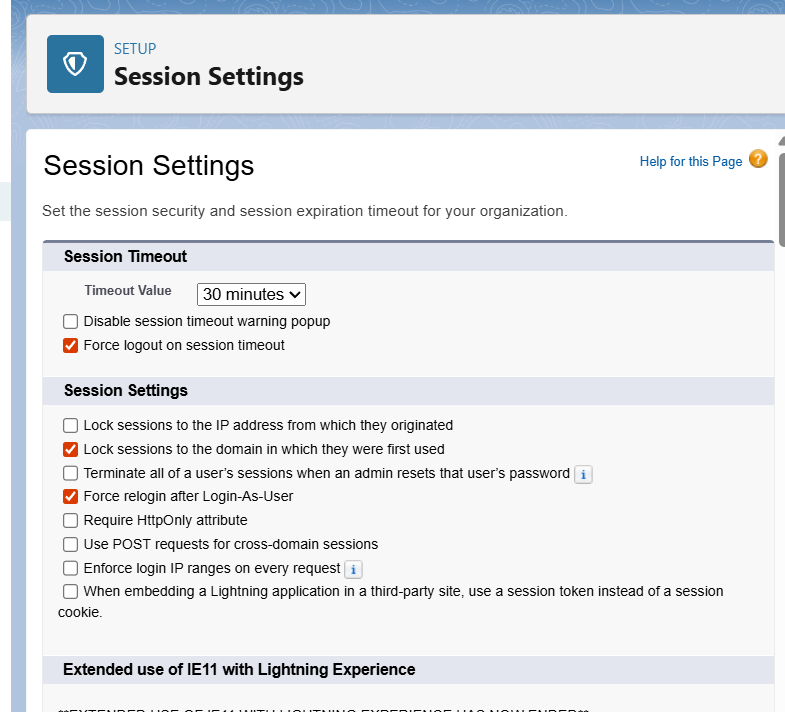
## 6) Session Settings (30 minute timeout)

Steps:

1) Setup → Security → Session Settings → Session timeout.

2) Set Timeout to 30 minutes. Optionally require reauthentication for sensitive pages.

3) Test by logging in as a test user and staying idle for 30 minutes to confirm session expiration and re-login behaviour.



## 7) Login IP Ranges — Restrict Agents to Office IP

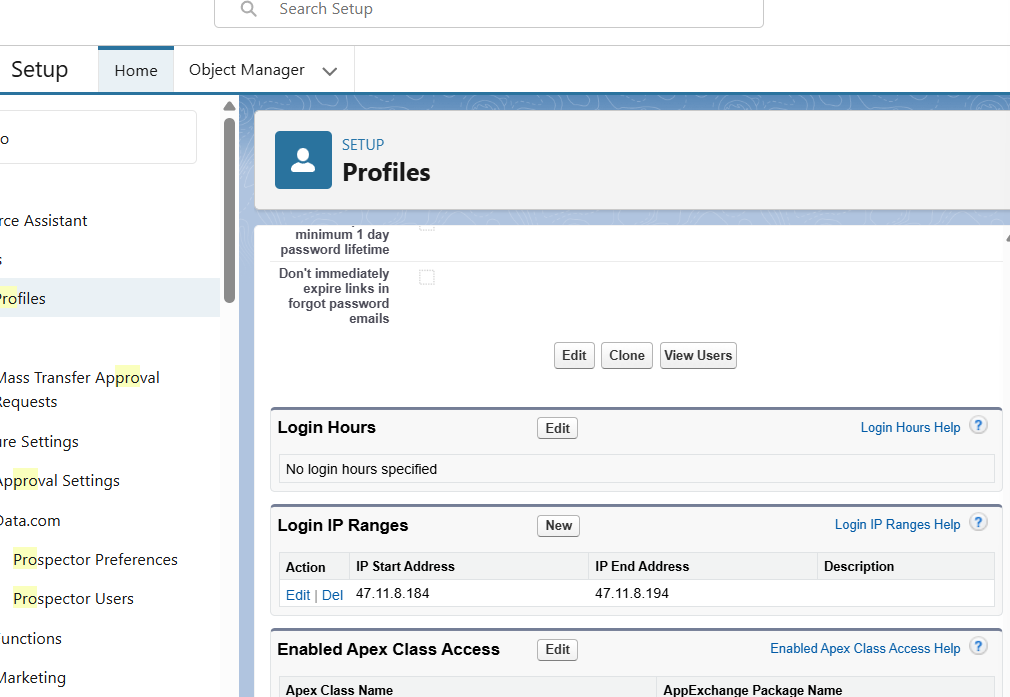
Approach: Add Login IP Ranges on the Agent profile so agents cannot login outside office IPs; keep a 'break glass' admin profile with broad access.

Steps:

1) Identify office public IP(s) — ask IT or use a 'what is my IP' tool from the office network.

2) Setup → Users → Profiles → Open 'Agent' profile → Login IP Ranges → New.

3) Enter Start IP and End IP (same if single IP). Save.

Warning: If you scope incorrectly you may lock out users. Always test with a test agent account and have administrators' IP or access exempted. 

## 8) Audit Trail & Field History Tracking

A. Field History Tracking (object-level):

1) Setup → Object Manager → Booking\_\_c → Fields & Relationships → Set History Tracking.

2) Select fields to track (e.g., Booking\_Status\_\_c, Start\_Date\_\_c, End\_Date\_\_c, Owner). Save.

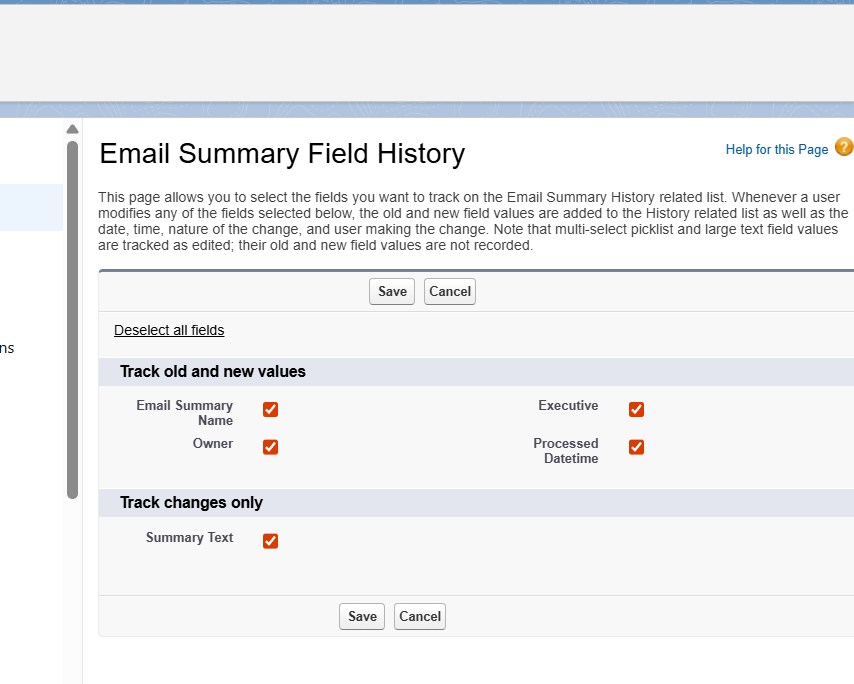
3) Access the 'Booking History' related list on Booking page layout or build a 'Booking Field History' report (Standard report type).

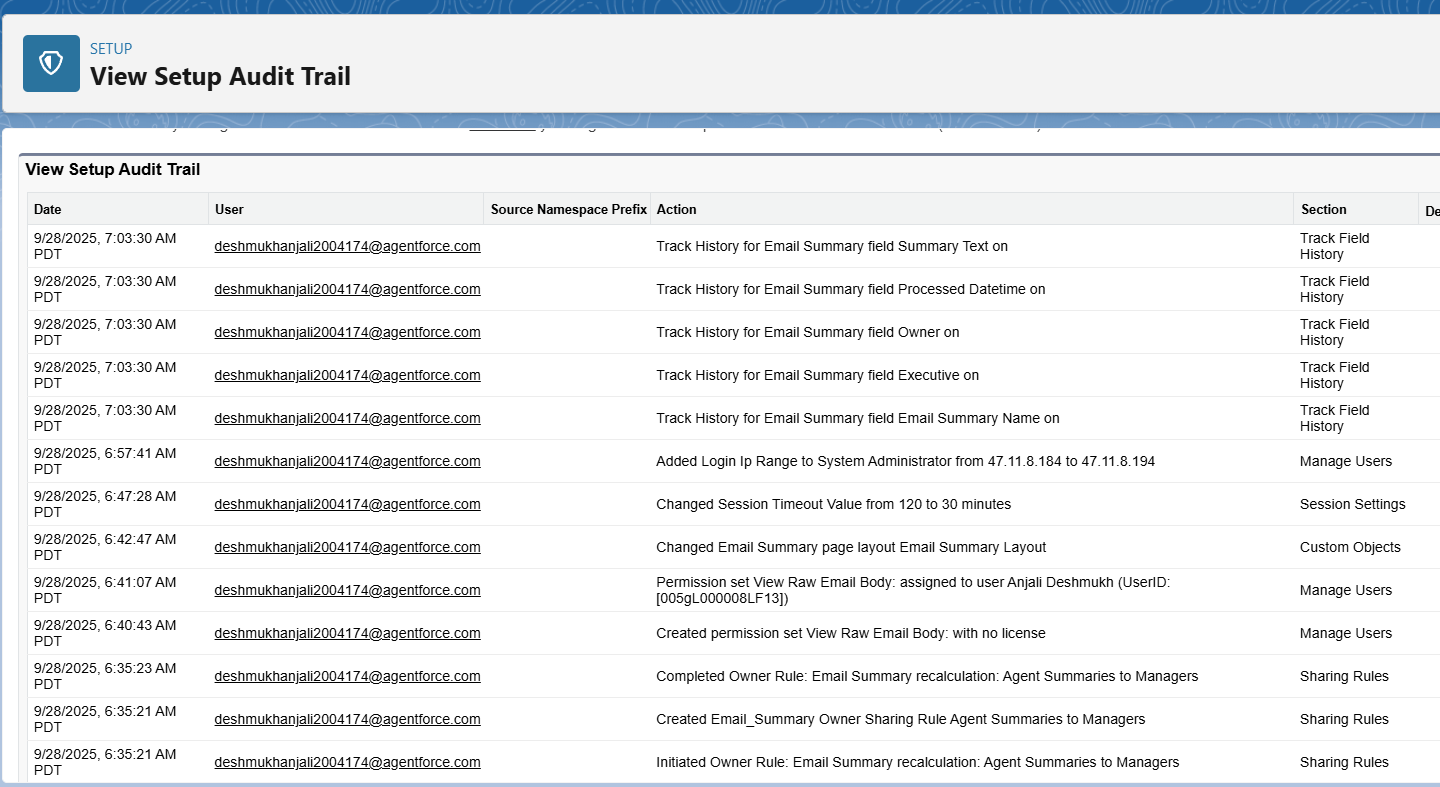
B. Setup Audit Trail (who changed setup):

Steps:

1) Setup → Security → View Setup Audit Trail. You can download the last 180 days of setup changes.

2) For longer retention consider 'Field Audit Trail' (paid add-on) or streaming Event Monitoring.





## 9) Testing, Validation & Acceptance

Create a simple test plan and execute it for each feature below. Sample test cases:

1) Reports: Create three bookings for Car A in date range → Run Cars Utilization → Verify SUM(Rental\_Days\_\_c) equals expected days and SUM(Total\_Revenue\_\_c) matches calculated amount.

2) Dynamic Dashboard: Login as agent\_test → View Fleet dashboard set as 'The dashboard viewer' → Verify only records owned by agent\_test appear.

3) Field Security: Login as an Agent → Open a Booking → Confirm 'Customer ID Proof' is not visible. Login as Manager → Confirm it is visible.

4) Login IP Ranges: Test logging in from office network (allowed) and from home (blocked).

5) Session Timeout: Remain idle for 30 minutes and confirm session expires.

6) Audit Trail: Change Booking\_Status\_\_c as Admin → Check Booking History and Setup Audit Trail entries.

**Purpose of Testing, Validation & Acceptance**

1. **Verify functionality** – Confirm that features like task creation, email summarization, dashboards, and reports behave according to design.
2. **Ensure data accuracy** – Check that calculations (e.g., totals, averages) in reports match expected values.
3. **Validate security and permissions** – Ensure sensitive fields are hidden from unauthorized users and record visibility rules are enforced.
4. **Confirm compliance** – Confirm login policies, session timeout, and audit trails are correctly implemented.
5. **Provide evidence for stakeholders** – A test plan shows managers, executives, or professors that the system has been rigorously considered even if full testing is not executed.

**Key Testing Areas and Theoretical Use Cases**

**1. Reports Validation**

**Objective:** Verify that reports calculate and display correct data.

**Theory:**

* Reports aggregate data from Salesforce objects using filters, grouping, and summary formulas.
* For example, a Cars Utilization report sums rental days and total revenue for a specific car. Accurate calculations confirm that report logic and data entry are correct.

**Use Case:**

* Create multiple bookings for a car → Run Cars Utilization report → The report should show total rental days and total revenue matching expected calculations.

**2. Dynamic Dashboards**

**Objective:** Ensure each user sees only the data they are authorized to view.

**Theory:**

* Dynamic dashboards use the viewer’s permissions and record-level security (OWD + sharing rules) to display personalized data.
* This prevents data leakage while providing the same layout to all users.

**Use Case:**

* An agent logs in → Views Fleet Dashboard → Should see only their bookings and metrics.
* A manager logs in → Views the same dashboard → Should see aggregated data for all agents in their team.

**3. Field-Level Security Validation**

**Objective:** Confirm sensitive fields are hidden from unauthorized users.

**Theory:**

* Salesforce uses Field-Level Security (FLS) to control visibility of sensitive data at the profile or permission set level.
* This ensures compliance with privacy policies and prevents accidental exposure of confidential information.

**Use Case:**

* Agent logs in → Opens a booking → Cannot see Customer ID Proof.
* Manager logs in → Can see Customer ID Proof for verification and oversight.

**4. Login IP Ranges**

**Objective:** Ensure users can access Salesforce only from authorized networks.

**Theory:**

* Restricting login by IP range enhances security by preventing unauthorized remote access.
* Office IP addresses are whitelisted; attempts to login from outside are blocked.

**Use Case:**

* Agent logs in from office → Allowed.
* Agent logs in from home or public Wi-Fi → Access denied.

**5. Session Timeout**

**Objective:** Confirm inactive users are automatically logged out.

**Theory:**

* Salesforce sessions expire after a set period of inactivity (e.g., 30 minutes) to prevent unauthorized access on unattended devices.
* Protects sensitive data from being exposed due to idle sessions.

**Use Case:**

* Agent logs in → Remains inactive for 30 minutes → Salesforce automatically logs out the session.

**6. Audit Trail Verification**

**Objective:** Track changes to critical fields and setup configurations.

**Theory:**

* Field History Tracking and Setup Audit Trail monitor who changed what, when, and from which user.
* Provides accountability and traceability, especially for sensitive operations like changing booking status or modifying executive summaries.

**Use Case:**

* Admin changes Booking\_Status\_\_c → Field History shows old and new status, user, and timestamp.
* Setup Audit Trail shows who modified security or configuration settings.

**Conclusion**

Even if this step is skipped for practical reasons, including a **theoretical explanation** in your documentation demonstrates:

1. Awareness of QA best practices.
2. Understanding of Salesforce security, reporting, and dashboard behavior.
3. Recognition of user-specific access, privacy, and compliance requirements.

This step ensures stakeholders know the project design is robust and capable of being validated if needed, without requiring full execution.