**Bank System**

**Insert**

1. The user fills out an insert form on the bank portal to open a new account, which is directly processed by the backend and stored in the database.
2. An employee uses the insert interface to approve a loan, requiring extra business logic before saving it in the database.
3. A customer submits a new request form via the frontend, and the backend handles validation before inserting into the bank's database.
4. The branch manager initiates a bulk account opening which involves extra approval layers before DB insertion.
5. A direct deposit instruction is submitted by the user, processed by backend logic, and inserted into the DB ledger.

**Delete**

1. The user deletes their saved payee list directly via the portal, triggering a backend call to delete the record.
2. An employee removes a dormant account after business verification and approval steps.
3. A customer request to delete a failed transaction is processed immediately by the system backend and database.
4. A manager deletes customer complaints from the admin dashboard after reviewing.
5. Fraudulent account data is flagged and deleted after compliance approval.

**Update**

1. A user directly edits their phone number through their online banking frontend, processed by backend and DB update.
2. A bank employee updates credit score data after an external report is received, needing business validation.
3. The customer updates the billing address via UI, backend checks format, and updates the record.
4. Account status is changed by support after internal ticket validation.
5. Admin updates interest rates which propagate through backend logic into financial tables.

**Query**

1. A customer views transaction history in real-time, querying from the database.
2. The finance department queries accounts with overdrafts, requiring secure backend access.
3. Users search for their last 10 transactions directly, served from backend cache or DB.
4. Manager runs reports on all new accounts each month.
5. The audit team runs compliance queries through restricted backend tools.

Perfect! Here's the next set of 20 sentences for the:

**Hospital System**

**Insert**

1. A patient uses the hospital frontend to book an appointment, which the backend validates and inserts into the appointments table.
2. An admin adds a new doctor to the system through a role-restricted interface, processed and saved in the doctor’s table.
3. A nurse enters a new patient’s vitals via the system, and the backend stores the data after validation.
4. A doctor prescribes a new medication through the medical record system, which then inserts the details into the prescriptions database.
5. A receptionist registers a walk-in patient directly via the UI, and the backend logic inserts the record into the patient’s table.

**Delete**

1. An admin deletes a canceled appointment through the hospital management panel, which removes it from the database.
2. A senior doctor requests deletion of outdated test results, approved by the admin before backend processing.
3. A patient removes their insurance record via the portal, and backend logic handles the deletion in the database.
4. A hospital manager deletes a duplicated staff entry after reviewing.
5. Compliance deletes expired medical documents after system-automated checks.

**Update**

1. A patient updates their contact information through the hospital portal, triggering backend logic to update the database.
2. A doctor modifies a diagnosis after re-evaluation, which is then updated in the patient’s medical history.
3. An admin adjusts staff schedules from the internal dashboard, and changes reflect in the DB after validation.
4. A receptionist corrects a typo in a patient’s name using the frontend form.
5. The system updates the room status from “Occupied” to “Available” after a discharge event.

**Query**

1. A patient logs in to view their upcoming appointments, which are queried from the appointments table.
2. A doctor checks a patient’s history using backend logic to retrieve data from various tables.
3. Admin staff query daily visit counts for hospital statistics.
4. Nurses check the availability of rooms through a live system query.
5. Management is running a report on patient admissions for the current month.

**Hotel System**

**Insert**

1. A guest books a room through the hotel website, and the backend inserts the reservation into the database.
2. A manager adds a new room type via the admin portal, and it is stored in the rooms table after validation.
3. The front desk staff registers a walk-in customer directly through the system, which processes and inserts the data.
4. A marketing team member creates a seasonal offer in the system, saved into the promotions database.
5. A customer uses the mobile app to add a special request to their booking, which is sent to the backend and inserted.

**Delete**

1. A user cancels their reservation using the frontend UI, triggering backend logic to delete it from the DB.
2. A receptionist removes an incorrect guest entry from the system after reviewing.
3. Management deletes outdated promotional codes from the database after a campaign ends.
4. A room that’s under renovation is temporarily removed from the booking pool via backend logic.
5. A hotel admin deletes reviews that violate content policy using a moderation interface.

**Update**

1. A guest updates check-in time via the hotel app, and backend logic applies the update.
2. A front desk clerk edits a guest's billing details after verifying identity.
3. The admin panel is used to change room rates for high season, which backend logic updates in the database.
4. Housekeeping updates room status from “Dirty” to “Clean” using a tablet interface.
5. A staff member corrects a guest’s name spelling in the booking record.

**Query**

1. A user searches for available rooms by date and budget, which triggers a backend query to the rooms database.
2. The hotel manager queries all current in-house guests from the admin dashboard.
3. A receptionist checks room availability in real-time before making a booking.
4. The analytics team runs a report on monthly occupancy rates.
5. A returning guest checks their past stay history through the app.

**Airline System**

**Insert**

1. A customer books a flight through the airline website, and the backend validates and inserts the reservation into the bookings database.
2. An airline admin adds a new flight schedule using the internal dashboard, which backend logic inserts into the flights table.
3. A check-in agent adds extra baggage details for a passenger, stored in the baggage records.
4. A frequent flyer enrolls in a loyalty program using the mobile app, triggering a backend insert into the loyalty database.
5. A flight crew member submits a delay report via the system, which gets logged into the operational records table.

**Delete**

1. A user cancels their flight booking via the frontend portal, and backend logic removes the record from the bookings table.
2. The airline operations team deleted a flight schedule that was entered by mistake.
3. A staff member removes a duplicate entry of a passenger record after backend validation.
4. An admin deletes old notifications from the customer service dashboard.
5. A safety officer deletes outdated incident reports that are no longer needed.

**Update**

1. A passenger updates their passport details via the airline website, and the backend reflects the changes in the customer database.
2. A flight coordinator adjusts departure time due to weather, and backend logic updates the flights table.
3. A gate agent modifies seat assignment through the internal system.
4. A user updates their meal preference before the flight, which is updated in the service request table.
5. The airline manager updates seasonal flight pricing in the pricing module, which syncs to the main DB.

**Query**

1. A customer searches for available flights using the frontend, which fetches real-time data from the flights database.
2. A staff member checks passenger manifest via the backend query tool.
3. A flight status checker retrieves current delays and arrival times for public view.
4. The system runs automated queries to list overbooked flights for staff review.
5. The analytics dashboard provides monthly data on flight occupancy and customer trends.

**E-Commerce System**

**Insert**

1. A customer adds an item to their cart via the website, which triggers backend logic to insert the item into the cart table.
2. A seller lists a new product using the seller dashboard, and the backend inserts the product details into the inventory database.
3. A customer places an order, and the backend logic creates a new order record in the orders table.
4. A marketing admin creates a new discount voucher, which is inserted into the promotions database.
5. A new user signs up on the platform, and the registration details are inserted into the user database after validation.

**Delete**

1. A customer removes an item from their shopping cart using the UI, and the system deletes it from the cart table.
2. A seller deletes a product listing that is out of stock or discontinued.
3. The admin deletes expired promotional offers from the system dashboard.
4. A user deletes their saved address from the account settings page.
5. A support agent deletes fake reviews flagged through a moderation process.

**Update**

1. A customer updates their delivery address via the account portal, and the backend updates the address table.
2. A seller updates the price and description of a product through their dashboard.
3. A customer changes the payment method on a pending order before checkout completion.
4. Admin staff update delivery policies and terms, which reflect in user views after backend updates.
5. An inventory manager updates stock quantities after a shipment arrives.

**Query**

1. A customer searches for products by category, price, and rating, which fetches filtered results from the database.
2. A seller queries their monthly sales report from the analytics dashboard.
3. The system queries product recommendations based on the user’s previous orders.
4. A customer tracks their order by querying its shipping status.
5. Admins generate financial reports by querying completed orders over the last quarter.

**YouTube System**

**Insert**

1. A user uploads a video via the platform, and the backend processes and inserts metadata into the videos table.
2. A creator adds a new playlist, which the backend inserts into the playlists database.
3. A viewer posts a comment on a video, and the system stores it after filtering for spam or violations.
4. A content manager adds subtitle tracks to a video using the editor, inserted into the subtitles database.
5. A user saves a video to the "Watch Later" list, which triggers an insert into their personal video list.

**Delete**

1. A creator deletes a video they no longer wish to keep online, and the backend removes it from the video database.
2. A viewer deletes their comment on a video, triggering backend logic to remove it.
3. YouTube’s moderation team deletes flagged content for violating community guidelines.
4. A user deletes a video from their "Liked Videos" list through the app.
5. An admin removes spam playlists using the internal moderation dashboard.

**Update**

1. A creator edits a video’s title and description, and the changes are updated in the videos table.
2. A user changes privacy settings from public to private, updated through backend logic.
3. A content manager updates tags and keywords for search optimization.
4. The system updates view counts, likes, and shares in real time as users interact.
5. A creator updates the monetization setting of their video from non-monetized to monetized.

**Query**

1. A viewer searches for videos by keyword, which triggers a backend search query across multiple indexed fields.
2. A user views their watch history by querying their personal playback log.
3. The platform recommends videos based on user activity using background queries.
4. Creator checks analytics for a video (watch time, demographics, etc.) queried from stats tables.
5. YouTube queries trending videos by region to populate the trending page dynamically.