Airbnb Sr. Manager, Advanced Analytics (Marketing) — Prep Pack

MySQL Interview Drills + Experimentation SOP • Generated 2025-10-24

A. Minimal Schema Snapshot

- users(user_id PK, created_at, country_code, platform, user_type)

- events(user_id, event_time, event_name, listing_id, device)

- bookings(booking_id PK, guest_id, host_id, listing_id, booked_at, status, nights, price_usd, marketing_channel)

- listings(listing_id PK, host_id, created_at, country_code, city, room_type)

- assignments(user_id, exp_id, variant, assigned_at)

- payments(payment_id PK, booking_id, paid_at, amount_usd, refunded_usd)

- marketing_spend(day, channel, country_code, spend_usd)

- reviews(review_id PK, booking_id, rating, created_at)

1) De-dupe latest row per email

```
-- Keep the most recent user row per email by updated_at
WITH ranked AS (
  SELECT
    u.*,
    ROW_NUMBER() OVER (
      PARTITION BY u.email ORDER BY u.updated_at DESC
    ) AS rn
  FROM users u
)
SELECT *
FROM ranked
WHERE rn = 1;
```

2) 30-minute sessionization from clickstream

```
-- Make session ids per user using 30-min inactivity rule
WITH ordered AS (
  SELECT
    e.*,
    LAG(e.event_time) OVER (
      PARTITION BY e.user_id ORDER BY e.event_time
    ) AS prev_ts
  FROM events e
),
flags AS (
  SELECT
    *,
    CASE
      WHEN prev_ts IS NULL THEN 1
```

```
      WHEN TIMESTAMPDIFF(MINUTE, prev_ts, event_time) > 30 THEN 1
      ELSE 0
    END AS is_new_session
  FROM ordered
),
numbered AS (
  SELECT
    *,
    SUM(is_new_session) OVER (
      PARTITION BY user_id
      ORDER BY event_time
      ROWS UNBOUNDED PRECEDING
    ) AS sess_num
  FROM flags
)
SELECT
  user_id,
  event_time,
  event_name,
  CONCAT(user_id, '-', sess_num) AS session_id
FROM numbered;
```

3) 3-step funnel (Search→View→Book) within 7 days

```
-- For each user, find earliest Search, then View after Search,
-- then Book within 7 days of Search
WITH first_step AS (
  SELECT user_id, MIN(event_time) AS t_search
  FROM events
  WHERE event_name = 'search'
  GROUP BY user_id
),
second_step AS (
  SELECT e.user_id, MIN(e.event_time) AS t_view
  FROM events e
  JOIN first_step s
    ON e.user_id = s.user_id
   AND e.event_name = 'view'
   AND e.event_time > s.t_search
   AND e.event_time <= s.t_search + INTERVAL 7 DAY
  GROUP BY e.user_id
),
third_step AS (
  SELECT b.guest_id AS user_id, MIN(b.booked_at) AS t_book
  FROM bookings b
  JOIN first_step s
    ON b.guest_id = s.user_id
   AND b.booked_at > s.t_search
   AND b.booked_at <= s.t_search + INTERVAL 7 DAY
  GROUP BY b.guest_id
)
SELECT
  COUNT(*) AS users_with_search,
  COUNT(second_step.user_id) AS search_to_view,
  COUNT(third_step.user_id) AS search_to_book,
  ROUND(
    COUNT(second_step.user_id) / NULLIF(COUNT(*), 0), 4
  ) AS view_rate,
  ROUND(
    COUNT(third_step.user_id) / NULLIF(COUNT(*), 0), 4
  ) AS book_rate
FROM first_step
LEFT JOIN second_step USING (user_id)
```

```
LEFT JOIN third_step USING (user_id);
```

4) Weekly signup cohorts and W+4 retention

```
-- Cohort week is Monday-based start-of-week
WITH base AS (
  SELECT
    u.user_id,
    DATE_SUB(DATE(u.created_at),
      INTERVAL WEEKDAY(DATE(u.created_at)) DAY) AS cohort_week
  FROM users u
),
activity AS (
  SELECT DISTINCT
    e.user_id,
    DATE_SUB(DATE(e.event_time),
      INTERVAL WEEKDAY(DATE(e.event_time)) DAY) AS active_week
  FROM events e
)
SELECT
  b.cohort_week,
  COUNT(DISTINCT b.user_id) AS cohort_size,
  COUNT(DISTINCT CASE
    WHEN a.active_week = b.cohort_week + INTERVAL 4 WEEK
    THEN b.user_id END) AS retained_w4,
  ROUND(
    COUNT(DISTINCT CASE
      WHEN a.active_week = b.cohort_week + INTERVAL 4 WEEK
    THEN b.user_id END)
    / NULLIF(COUNT(DISTINCT b.user_id), 0), 4
  ) AS w4_retention
FROM base b
LEFT JOIN activity a
  ON a.user_id = b.user_id
GROUP BY b.cohort_week
ORDER BY b.cohort_week;
```

5) 28-day rolling GMV by country (daily grains)

```
-- Daily GMV then a 28-day rolling sum using a correlated range filter
WITH daily AS (
  SELECT
    DATE(b.booked_at) AS day,
    l.country_code,
    SUM(b.price_usd) AS gmv
  FROM bookings b
  JOIN listings l ON l.listing_id = b.listing_id
  WHERE b.status = 'booked'
  GROUP BY DATE(b.booked_at), l.country_code
)
SELECT
  d1.day,
  d1.country_code,
  d1.gmv AS gmv_day,
  (
    SELECT SUM(d2.gmv)
    FROM daily d2
    WHERE d2.country_code = d1.country_code
      AND d2.day BETWEEN d1.day - INTERVAL 27 DAY AND d1.day
  ) AS gmv_28d
FROM daily d1
ORDER BY d1.country_code, d1.day;
```

6) P90 host response time (rank-based percentile)

```
-- Given table 'inquiries(host_id, responded_sec)'
WITH ranked AS (
  SELECT
    host_id,
    responded_sec,
    ROW_NUMBER() OVER (
      PARTITION BY host_id ORDER BY responded_sec
    ) AS rn,
    COUNT(*) OVER (PARTITION BY host_id) AS cnt
  FROM inquiries
)
SELECT
  host_id,
  responded_sec AS p90_responded_sec
FROM ranked
WHERE rn = CEIL(0.90 * cnt);
```

7) SRM check: treatment share vs 95% CI

```
-- Compute daily treatment share, 95% CI, and SRM flag
WITH counts AS (
  SELECT
    DATE(assigned_at) AS day,
    SUM(variant = 'treatment') AS n_treat,
    COUNT(*) AS n_total
  FROM assignments
  WHERE exp_id = 101
  GROUP BY DATE(assigned_at)
)
SELECT
  day,
  n_treat,
  n_total,
  ROUND(n_treat / NULLIF(n_total, 0), 4) AS p_hat,
  -- Wald 95% CI (ok for quick SRM triage at decent n)
  ROUND(
    1.96 * SQRT(
      (n_treat / NULLIF(n_total,0))
      * (1 - (n_treat / NULLIF(n_total,0)))
      / NULLIF(n_total,0)
    ), 4
  ) AS margin,
  CASE
    WHEN ABS((n_treat / NULLIF(n_total,0)) - 0.5) >
         1.96 * SQRT(0.5 * 0.5 / NULLIF(n_total,0))
    THEN 1 ELSE 0
  END AS srm_flag
FROM counts
ORDER BY day;
```

8) Variant lift by segment with z-score

```
-- Conversion by variant within segment, z across variants
WITH conv AS (
  SELECT
    s.segment, a.variant,
    SUM(b.booking_id IS NOT NULL) AS conv,
    COUNT(*) AS n
  FROM assignments a
  JOIN users u ON u.user_id = a.user_id
```

```
  JOIN (
    SELECT user_id,
           CASE
             WHEN platform IN ('ios','android') THEN 'mobile'
             ELSE 'desktop'
           END AS segment
    FROM users
  ) s ON s.user_id = a.user_id
  LEFT JOIN bookings b
    ON b.guest_id = a.user_id
   AND b.booked_at BETWEEN a.assigned_at
                       AND a.assigned_at + INTERVAL 14 DAY
  WHERE a.exp_id = 101
  GROUP BY s.segment, a.variant
),
pivot AS (
  SELECT
    segment,
    MAX(CASE WHEN variant='control' THEN conv END) AS c_conv,
    MAX(CASE WHEN variant='control' THEN n    END) AS c_n,
    MAX(CASE WHEN variant='treatment' THEN conv END) AS t_conv,
    MAX(CASE WHEN variant='treatment' THEN n    END) AS t_n
  FROM conv
  GROUP BY segment
)
SELECT
  segment,
  ROUND(t_conv / NULLIF(t_n,0), 4) AS p_t,
  ROUND(c_conv / NULLIF(c_n,0), 4) AS p_c,
  ROUND(
    (t_conv / NULLIF(t_n,0)) - (c_conv / NULLIF(c_n,0)), 4
  ) AS lift,
  -- Pooled standard error & z
  ROUND(
    (
      (t_conv / NULLIF(t_n,0)) - (c_conv / NULLIF(c_n,0))
    ) / SQRT(
      ((t_conv + c_conv) / NULLIF(t_n + c_n,0))
      * (1 - ((t_conv + c_conv) / NULLIF(t_n + c_n,0)))
      * (1/NULLIF(t_n,0) + 1/NULLIF(c_n,0))
    ), 3
  ) AS z_score
FROM pivot;
```

9) Mix-shift vs within-segment AOV decomposition

```
-- Periods P0 and P1, category = room_type, AOV = revenue/orders
WITH order_cat AS (
  SELECT
    CASE
      WHEN DATE(booked_at) < '2025-07-01' THEN 'P0' ELSE 'P1'
    END AS period,
    l.room_type AS cat,
    COUNT(*) AS orders,
    SUM(price_usd) AS rev
  FROM bookings b
  JOIN listings l ON l.listing_id = b.listing_id
  WHERE b.status='booked'
  GROUP BY period, l.room_type
),
agg AS (
  SELECT
    period,
```

```
    SUM(orders) AS orders,
    SUM(rev) AS rev
  FROM order_cat
  GROUP BY period
),
shares AS (
  SELECT
    oc.*,
    oc.orders / NULLIF(a.orders,0) AS w,
    oc.rev / NULLIF(oc.orders,0) AS aov_cat,
    a.rev / NULLIF(a.orders,0) AS aov_all
  FROM order_cat oc
  JOIN agg a USING (period)
),
decomp AS (
  SELECT
    -- Mix effect: change in weights applied to baseline cat AOVs
    SUM(CASE WHEN period='P1' THEN w END
        * CASE WHEN period='P0' THEN aov_cat END) AS mix_on_p0,
    -- Within effect: change in cat AOVs applied to baseline weights
    SUM(CASE WHEN period='P0' THEN w END
        * CASE WHEN period='P1' THEN aov_cat END) AS within_on_p0,
    MAX(CASE WHEN period='P0' THEN aov_all END) AS aov_p0,
    MAX(CASE WHEN period='P1' THEN aov_all END) AS aov_p1
  FROM shares
)
SELECT
  aov_p0,
  aov_p1,
  (within_on_p0 - aov_p0) AS within_effect,
  (mix_on_p0    - aov_p0) AS mix_effect,
  (aov_p1 - aov_p0)       AS total_change
FROM decomp;
```

10) Convert UTC event timestamps to local day

```
-- MySQL requires time zone tables loaded for named zones
SELECT
  user_id,
  event_time AS ts_utc,
  CONVERT_TZ(event_time, 'UTC', 'America/Los_Angeles') AS ts_pt,
  DATE(CONVERT_TZ(event_time, 'UTC', 'America/Los_Angeles')) AS day_pt
FROM events;
```

11) Cancel-rate guardrail by variant/day

```
WITH by_day AS (
  SELECT
    DATE(b.booked_at) AS day,
    a.variant,
    SUM(b.status='cancelled') AS cancels,
    SUM(b.status='booked') AS booked
  FROM assignments a
  JOIN bookings b
    ON b.guest_id = a.user_id
   AND b.booked_at BETWEEN a.assigned_at
                       AND a.assigned_at + INTERVAL 14 DAY
  WHERE a.exp_id = 101
  GROUP BY DATE(b.booked_at), a.variant
)
SELECT
  day,
```

```
  variant,
  cancels,
  booked,
  ROUND(cancels / NULLIF(booked + cancels,0), 4) AS cancel_rate,
  CASE WHEN (cancels / NULLIF(booked + cancels,0)) > 0.06
       THEN 1 ELSE 0 END AS breach
FROM by_day
ORDER BY day, variant;
```

12) CUPED in SQL (θ and adjusted outcome)

```
-- Y = bookings in test window; X = preperiod bookings (covariate)
WITH per_user AS (
  SELECT
    a.user_id,
    a.variant,
    SUM(CASE
      WHEN b.booked_at BETWEEN a.assigned_at
                           AND a.assigned_at + INTERVAL 14 DAY
      THEN 1 ELSE 0 END) AS Y,
    SUM(CASE
      WHEN b.booked_at BETWEEN a.assigned_at - INTERVAL 28 DAY
                           AND a.assigned_at
      THEN 1 ELSE 0 END) AS X
  FROM assignments a
  LEFT JOIN bookings b ON b.guest_id = a.user_id
  WHERE a.exp_id = 101
  GROUP BY a.user_id, a.variant
),
moments AS (
  SELECT
    AVG(X) AS EX,
    AVG(Y) AS EY,
    AVG(X*Y) AS EXY,
    AVG(X*X) AS EX2
  FROM per_user
),
theta AS (
  SELECT
    (EXY - EX*EY) / NULLIF(EX2 - EX*EX, 0) AS theta
  FROM moments
),
adjusted AS (
  SELECT
    p.user_id,
    p.variant,
    p.Y - t.theta * (p.X - m.EX) AS Y_star
  FROM per_user p
  CROSS JOIN moments m
  CROSS JOIN theta t
)
SELECT
  variant,
  AVG(Y) AS mean_Y,
  AVG(Y_star) AS mean_Y_star
FROM per_user
JOIN adjusted USING (user_id, variant)
GROUP BY variant;
```

13) Sample size per arm for two-proportion test (80% power)

```
-- Inputs: baseline p0, desired absolute MDE (delta)
```

```
WITH inputs AS (
  SELECT 0.08 AS p0, 0.01 AS delta, 0.84 AS z_beta, 1.96 AS z_alpha
),
calc AS (
  SELECT
    p0,
    p0 + delta AS p1,
    z_beta, z_alpha,
    (z_alpha*SQRT(2*p0*(1-p0)) + z_beta*
     SQRT(p0*(1-p0) + (p0+delta)*(1-(p0+delta)))) AS num
  FROM inputs
)
SELECT
  CEIL( (num*num) / (delta*delta) ) AS n_per_arm
FROM calc;
```

14) Late-arriving event de-dup by event_id

```
-- events_raw(event_id, event_time, ingest_time, ... )
WITH ranked AS (
  SELECT
    *,
    ROW_NUMBER() OVER (
      PARTITION BY event_id ORDER BY ingest_time DESC
    ) AS rn
  FROM events_raw
)
SELECT *
FROM ranked
WHERE rn = 1;
```

15) Hosts with zero bookings in the last 90 days

```
SELECT
  h.host_id
FROM hosts h
LEFT JOIN bookings b
  ON b.host_id = h.host_id
 AND b.booked_at >= CURRENT_DATE - INTERVAL 90 DAY
 AND b.status = 'booked'
GROUP BY h.host_id
HAVING COUNT(b.booking_id) = 0;
```

16) Reactivation: churned hosts returning this quarter

```
-- Churned = no bookings in prior 90 days; reactivated = booking now
WITH last_booking AS (
  SELECT host_id, MAX(booked_at) AS last_booked_at
  FROM bookings
  WHERE status='booked'
  GROUP BY host_id
),
churned AS (
  SELECT host_id
  FROM last_booking
  WHERE last_booked_at < CURRENT_DATE - INTERVAL 90 DAY
),
reactivated AS (
  SELECT DISTINCT host_id
  FROM bookings
  WHERE status='booked'
    AND booked_at >= DATE_FORMAT(CURRENT_DATE, '%Y-%m-01')
```

```
)
SELECT
  COUNT(*) AS churned_hosts,
  SUM(host_id IN (SELECT host_id FROM reactivated)) AS reactivated,
  ROUND(
    SUM(host_id IN (SELECT host_id FROM reactivated))
    / NULLIF(COUNT(*),0), 4
  ) AS reactivation_rate
FROM churned;
```

17) Top-K hosts by 30-day GMV with ties handled

```
WITH gmv AS (
  SELECT
    b.host_id,
    SUM(b.price_usd) AS gmv_30d
  FROM bookings b
  WHERE b.status='booked'
    AND b.booked_at >= CURRENT_DATE - INTERVAL 30 DAY
  GROUP BY b.host_id
),
ranked AS (
  SELECT
    host_id,
    gmv_30d,
    DENSE_RANK() OVER (ORDER BY gmv_30d DESC) AS rnk
  FROM gmv
)
SELECT host_id, gmv_30d
FROM ranked
WHERE rnk <= 10
ORDER BY gmv_30d DESC, host_id;
```

18) First booking within 28 days since signup (activation)

```
WITH first_booking AS (
  SELECT
    u.user_id,
    u.created_at,
    MIN(b.booked_at) AS first_booked_at
  FROM users u
  LEFT JOIN bookings b
    ON b.guest_id = u.user_id
  GROUP BY u.user_id, u.created_at
)
SELECT
  COUNT(*) AS signups,
  SUM(first_booked_at IS NOT NULL
      AND first_booked_at <= created_at + INTERVAL 28 DAY) AS activated,
  ROUND(
    SUM(first_booked_at IS NOT NULL
        AND first_booked_at <= created_at + INTERVAL 28 DAY)
      / NULLIF(COUNT(*),0), 4
  ) AS activation_rate_28d
FROM first_booking;
```

19) CAC by channel (daily)

```
-- New guests attributed to channel by first booking's marketing_channel
WITH first_book AS (
  SELECT
    guest_id,
```

```
    MIN(booked_at) AS first_booked_at
  FROM bookings
  WHERE status='booked'
  GROUP BY guest_id
),
daily_new AS (
  SELECT
    DATE(b.booked_at) AS day,
    b.marketing_channel AS channel,
    COUNT(*) AS new_guests
  FROM bookings b
  JOIN first_book fb
    ON fb.guest_id = b.guest_id
   AND fb.first_booked_at = b.booked_at
  GROUP BY DATE(b.booked_at), b.marketing_channel
)
SELECT
  ms.day,
  ms.channel,
  ms.spend_usd,
  COALESCE(dn.new_guests, 0) AS new_guests,
  ROUND(ms.spend_usd / NULLIF(dn.new_guests,0), 2) AS cac
FROM marketing_spend ms
LEFT JOIN daily_new dn
  ON dn.day = ms.day
 AND dn.channel = ms.channel
ORDER BY ms.day, ms.channel;
```

20) Chi-square SRM statistic (2-arm) for the day

```
WITH counts AS (
  SELECT
    DATE(assigned_at) AS day,
    SUM(variant='control')   AS n_c,
    SUM(variant='treatment') AS n_t
  FROM assignments
  WHERE exp_id = 101
  GROUP BY DATE(assigned_at)
),
stats AS (
  SELECT
    day, n_c, n_t, (n_c + n_t) AS n,
    (n_c - (n_c + n_t)/2.0) AS diff_from_exp
  FROM counts
)
SELECT
  day,
  n_c, n_t, n,
  -- Pearson chi-square for 1 df against 50/50 expectation
  ROUND( (4 * diff_from_exp * diff_from_exp) / NULLIF(n,0), 3 )
    AS chi2,
  -- For quick triage: chi2 > 3.84 ≈ p < 0.05
  CASE WHEN (4 * diff_from_exp * diff_from_exp) / NULLIF(n,0) > 3.84
       THEN 1 ELSE 0 END AS srm_flag
FROM stats
ORDER BY day;
```

C. Experimentation Guardrails, SRM, CUPED — One-Pager

Purpose

Protect users and the business while enabling fast, correct decisions from experiments and quasi-experiments.

Scope

All user-facing tests and major marketing/product initiatives; applies to allocation systems (experimentation, holdouts, geo tests).

1) Guardrails (global, “never-worse-than”)

- Canonical list (owned by Analytics; versioned): conversion, cancellations, refund rate, CSAT/review mean, response latency, supply/host activeness, top-of-funnel health, policy-sensitive metrics. - Thresholds: set from historical variance and risk. Example: `cancel_rate_treat - cancel_rate_ctrl` ≤ +0.3pp; `p90_latency` ≤ control + 5%. - Monitoring: daily readouts with 95% CIs, trend deltas, breach flags; auto-page for SRM or >2× breach. - Freeze at start; changes require change control ticket.

2) SRM (Sample Ratio Mismatch)

- Observed assignment share deviates beyond chance from expected split. - Checks: overall and by strata (device, locale, app version, cohort). Chi-square $p<0.05$ alert; $p<0.01$ auto-stop. - Triage: eligibility bias; collisions/sticky bucketing; ramp asymmetry; SDK version; data issues. - Action: if unresolved in 24h, pause and relaunch; do not read SRM-tainted results.

3) Primary outcomes & analysis

- Primary: 1–2 KPIs (e.g., book_rate_14d). Secondary: supporting behavior/quality. - Heterogeneity: pre-declared segments + exploratory with multiple-testing control. - Readout: effect size, 95% CI, power, mechanism sanity checks.

4) Variance reduction (CUPED)

- Use when stable pre-period covariate correlates with outcome ($\rho \geq 0.2$) and is pre-randomization. - Compute $\theta = \mathrm{Cov}(Y,X)/\mathrm{Var}(X)$; adjusted $Y^* = Y - \theta (X - E[X])$. - Governance: publish covariate; validate neutrality; log θ and variance reduction. - Benefit: ~10–40% power/speed gain; verify via backtest.

5) Decision policy

- Ship: primary improves ($p<0.05$ or posterior ≥95%); no guardrail breach; coherent mechanism; ≥MBR. - Don’t ship: SRM; guardrail breach; contradictory mechanism; large negative tail. - Iterate: if neutral, run faster follow-ups (targeted segment, stronger treatment, or variance reduction).

6) Ownership & artifacts

- Analytics: metric contracts, readouts, SRM/guardrail monitors. - Eng/PM: experiment switch, rollouts, eligibility. - Artifacts: pre-reg doc (hypothesis, metrics, MDE/power), launch ticket, Knowledge-Repo post (code + reproducible SQL), Superset dashboard with frozen definitions.