1. Head چیست؟
   1. یه رفرنس هست به آخرین checkout commit. هر وقت که ما یه کامیت میزنیم یا checkout ای انجام میدیم (اگه روی برنچ باشه به آخرین کامیت اون برنچ اگه هم یه کامیت خاص باشه که همون کامیت)، این head به اون کامیت مورد نظر اشاره میکنه.
2. Detached head چیست؟
   1. اگه به جای checkout کردن روی یه برنچ، روی یه کامیت checkout کنیم، اونوقت detached head داریم. این حالت ممکنه توی rebase هم بوجود بیاد.
   2. اینکار ایراداتی داره. مثلا اینکه ممکنه تغییرات جدید از بین بره چون میخوایم کامیت های جدید روی یه history commit بزنیم و اگه روی یه برنچ checkout کنیم ممکنه اون کامیت های جدید دیگه قابل دسترسی نباشن چون به هیچ برنچی رفرنس ندارن. از طرفی توی خود کار هایی که داریم انجام میدیم ممکنه سردرگمی پیش بیاد.
   3. برای درست کردن این مشکل، اول باید status رو چک کنیم که مطمئن بشیم در وضعیت detached head هستیم. بعدش میتونیم یه برنچ جدید بسازیم تا حداقل تغییرات از دست نرن. حالا میتونیم اگه میخوایم تغییر خاصی بدیم، انجام بدیم و بعدش تغییرات روی این برنج رو به برنچ مورد نظر منتقل کنیم.
3. Staging area و unstaging چیست؟
   1. هر وقت ما تغییراتی توی repository مون میدیم، قبل از اینکه این تغییرات بخواد کامیت بشه باید تایید بشن. یه index وجود داره که میاد فایل هایی که میخوایم تغییراتشون توی کامیت بعدی ثبت بشه رو نگه میداره توی یه staging area.
   2. حالا اگه بخوایم یه سری فایل ها (که تغییرشون دادیم و توی repository هم add شدن قبلا) رو توی کامیت بعدی لحاظ نکنیم، باید اونا رو unstage کنیم. (اینکارو با هر کدوم از دستورات reset و restore میشه انجام داد.
4. تفاوت merge و rebase چیست؟
   1. توی merge، میایم یه برنچ رو روی یه برنچ دیگه ترکیب می کنیم اما با یه commit جدید که این commit در واقع میشه ترکیب دو برنچ. اینجا ما هیستوری کامل رو داریم و دقیق تغییرات رو به همون صورتی که بوده برامون نگه میداره.  
      اما rebase میاد به جای یه commit جدید، کل اون commit هایی که توی برنج فرعی بوده رو روی برنچ اصلی بازنویسی می کنه. انگار که تغییرات برنج فرعی از اول داشته روی برنچ اصلی انجام میشده.
   2. توی merge ما هیستوری کامل و دقیق تغییرات رو داریم و به راحتی میتونیم conflict ها رو پیدا و رفع کنیم. اما همین هیستوری کامل ممکنه دردسر ساز بشه و گشتن بین commit ها سخت و طولانی تر بشه  
      توی rebase ما هیستوری خطی داریم که تعداد commit کمتر و تمیزتره اما هیستوری داره بازنویسی میشه و از طرفی اگه در حین rebase به conflict بخوریم، پیدا و رفع کردنش دردسر داره.
5. تفاوت fetch و pull چیست؟
   1. توی fetch ما آخرین اطلاعات رو از سرور میگیریم و روی لوکال داریمشون اما این تغییرات روی لوکال ما integrate نمیشه. مثل این میمونه که فقط یه اعلان وضعیت کنیم.
   2. اما توی pull، عملا دستور fetch به علاوه merge داره اجرا میشه؛ یعنی علاوه بر دریافت آخرین تغییرات از سرور، میاد اون تغییرات رو روی لوکال هم ترتیب اثر میده در نتیجه ما توی لاگ لوکال به علاوه برنچ ها و کامیت ها میتونیم تغییرات رو ببینیم.
6. Cherry pick چیست؟
   1. اینکه یه کامیت از یه برنچ دیگه رو روی یه برنچ متفاوت اعمال کنیم. مثلا دوتا برنچ a و b داریم. روی برنچ b یه سری تغییرات و کامیت های مختلف زدیم و الان میخوایم **بعضی** از اون کامیت ها رو ببریم روی برنچ a. اول checkout میکنیم روی برنچ a و بعدش میایم با دستور cherry-pick و کد hash اون کامیت ها، این کار رو انجام میدیم. این دستور میاد یه کامیت جدید روی برنچ a درست میکنه که تغییرات اون کامیتی که hash اش رو استفاده کردیم میذاره توی این کامیت جدید. اگه نخوایم کامیت رو انجام بده و صرفا تغییرات بیاد روی working tree، کافیه یه -n قبل از hash توی کامند بذاریم.
   2. این دستور برای وقتی که روی یه برنچ کامیت اشتباه زدیم و میخوایم ببریمش روی برنچ درست به کار میاد؛ یا وقتی که یه باگ پیش اومده و میخوایم برطرفش کنیم یا به سرعت یه کامیت که درست کار میکرده تا قبل از باگ رو استفاده کنیم (استفاده از آخرین نسخه قابل اجرا یا پایدار)
7. دستور amend چیست؟
   1. برای اعمال تغییر روی کامیت قبلی به کار میره بدون اینکه بخوایم کامیت جدیدی ایجاد کنیم.
8. تفاوت reset و revert چیست؟
   1. دستور reset برای تغییر توی هیستوری git و unstage کردن فایل ها به کار میره. برای استفاده اول، اینطوریه که Head رو میبره روی یه کامیت خاص و هر چی از اون کامیت به بعد هست رو پاک میکنه. برای استفاده دوم، صرفا اون فایل (ها) توی کامیت بعدی لحاظ نمیشه اما هنوز تغییرات داخلش، سرجاشون موندن. این دستور روی local repository اعمال میشه و میتونه روی کامیت، فایل یا برنچ اعمال بشه.
   2. دستور revert برای پاک کردن کامیت از روی remote استفاده میشه. یعنی یه کامیت رو از روی remote پاک میکنه اما خودش یه کامیت جدید اضافه میکنه که نشون میده اون یکی کامیت پاک شده. این دستور فقط روی remote و روی کامیت اعمال میشه.
9. upstream و downstream چیست؟
   1. ما یه ریپو local داریم و یه remote. هر کدوم داخل خودشون یه برنچ main دارن. اما برای اتصال بین این دوتا، یه برنج سومی داخل local داریم به اسم origin/main. این برنچ میشه دلال ارتباطی بین دوتا برنچ main مون. پس در واقع سه تا برنچ توی کل local و remote مون داریم.
   2. اگه بخوایم تغییرات رو از remote بگیریم (pull)، باید اول تغییرات بیان روی origin/main و بعدش از این جا میان روی main داخل local. این روند میشه downstream.
   3. اگه بخوایم تغییرات رو از local بزاریم روی remote (push) باید تغییرات بره روی origin/main و بعدش از این جا بره روی main توی remote. این روند میشه upstream.