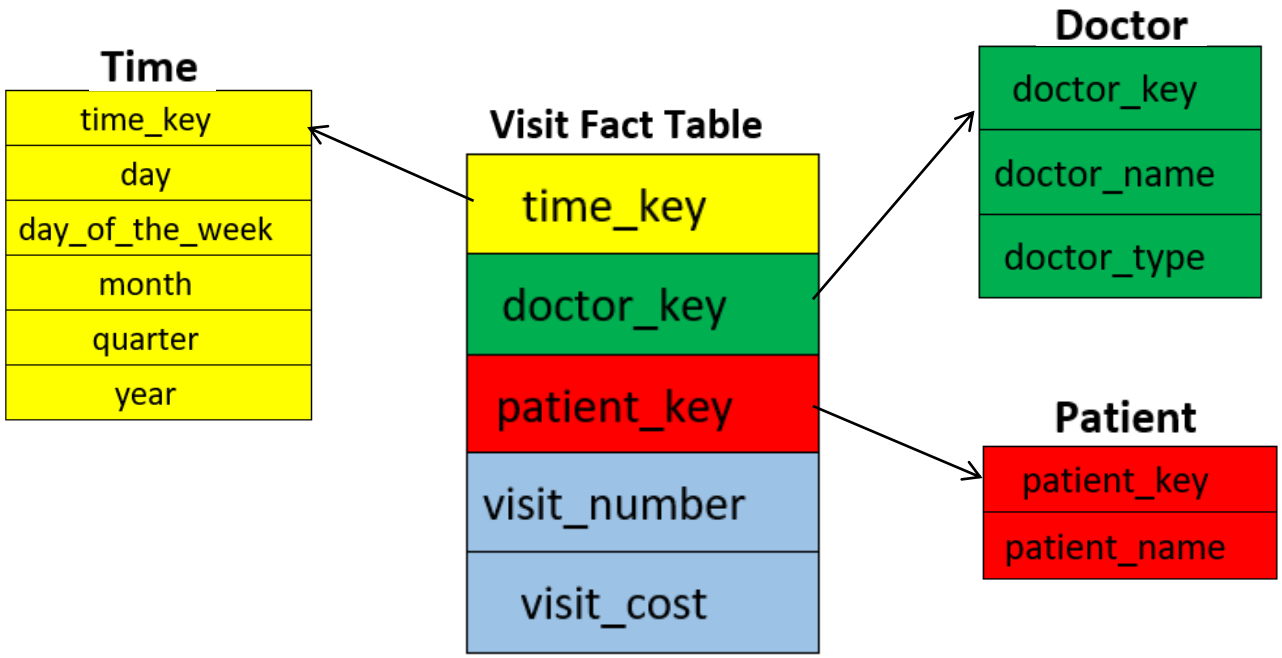
به نام خدا

مرضیه علیدادی – ۸۱۰۱۰۱۲۳۶

تکلیف ۲ داده کاوی

سوال ۱:

الف)



ب) مراحل مورد نیاز برای محاسبه‌ی مجموع درآمد هر پزشک در سال ۲۰۲۲ با شروع از cuboid پایه [patient, doctor, day]:

1. Roll-up on Time (from day to month)
2. Roll-up on Time (from month to quarter)
3. Roll-up on Time (from quarter to year)
4. Slice for (year = "2022")
5. Roll-up on Patient (from patient to all)

ج) مراحل مورد نیاز برای محاسبه‌ی مجموع هزینه‌ی هر بیمار بابت ویزیت شدن توسط دندانپزشک‌ها در سال ۲۰۲۲ با شروع از cuboid پایه [patient, doctor, day]:

1. Roll-up on Time (from day to month)
2. Roll-up on Time (from month to quarter)
3. Roll-up on Time (from quarter to year)
4. Dice for (year = "2022") and (doctor_type = "dentist")
5. Roll-up on Doctor (from doctor to all)

سوال ۲:

- مجموع ۳۰ مقدار بیشینه: algebraic

می‌توان با استفاده از تابع max_N()، با قرار دادن N برابر با ۳۰، ۳۰ مقدار بیشینه را پیدا کرد. این تابع که از نوع algebraic است، N بار از تابع max که از نوع distributive است، استفاده می‌کند. هر بار بیشترین مقدار را پیدا می‌کند و آن را حذف می‌کند و دوباره با استفاده از تابع max، بیشترین مقدار بین بقیه‌ی مقادیر باقی مانده را پیدا می‌کند. و همین روند تا پیدا کردن همه‌ی ۳۰ مقدار بیشینه، ادامه می‌دهد. سپس، با استفاده از تابع sum() که از نوع distributive است، مجموع این ۳۰ مقدار یافت شده محاسبه می‌شود.

با توجه به روندی که برای محاسبه‌ی این تابع استفاده شد، با استفاده از یک سری توابع distributive، توانستیم آن را محاسبه کنیم. در نتیجه، این تابع از نوع algebraic است.

- مُد، درصورتی که داده‌ها از نوع باینری باشند: algebraic

مُد، در حالت کلی و روی داده‌های general، به این شکل قابل محاسبه نیست که بتوان داده‌ها را به چند دسته تقسیم کرد و با اجرای آن بر روی این بخش‌های کوچک‌تر، مُد کل داده‌ها را محاسبه کرد. و همچنین نمی‌توان از توابع distributive دیگر کمک گرفت و مُد کل داده‌ها را محاسبه کرد. درنتیجه، holistic است.

اینجا، در حالتی که داده‌ها از نوع باینری هستند، با یک مثال نشان داده می‌شود که همچنان نمی‌توان از اجرای مُد بر روی بخش‌های کوچک استفاده کرد و مُد کل داده‌ها را محاسبه کرد:

فرضاً داده‌ها عبارتند از:

1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0

و آن‌ها را بدین شکل بخش بندی می‌کنیم:

{1, 1, 0}, {1, 1, 0}, {1, 1, 0}, {0, 0, 0}, {0, 0, 0}

با اجرای تابع مُد روی هر یک از این بخش‌ها، مُد آن‌ها به ترتیب از چپ به راست، برابر ۱، ۱، ۱، ۰ و ۰ می‌شود. روشی که به ذهن می‌رسد این است که تعداد ۱ها و ۰ها را جداگانه با کمک تابع ()count بشماریم و بین حاصل این دو، ()max را محاسبه کنیم. و آن را به عنوان مُد کل داده‌ها در نظر بگیریم. اما با این روش، مقدار حاصل، برابر ۱ می‌شود؛ در صورتی که مشخص است مُد این داده‌ها برابر ۰ است. پس با وجود باینری بودن داده‌ها نیز، همچنان نمی‌توان از اجرای تابع مُد بر روی بخش‌های مختلف داده‌ها کمک گرفت و مُد کل داده‌ها را محاسبه کرد. پس تابع مُد بر روی داده‌های باینری، distributive نیست.

اما از توابع distributive می‌توان برای محاسبه‌ی آن کمک گرفت. بدین شکل که ابتدا از تابع ()sum استفاده می‌شود. حاصل این تابع، در واقع تعداد ۱ها را نشان می‌دهد. سپس از تابع ()count استفاده می‌شود، تا تعداد کل داده‌ها محاسبه شود. حاصل آن، تقسیم بر ۲ می‌شود. اگر تعداد

۱ها که با sum محاسبه شد، از این عدد کوچک‌تر بود، یعنی کم‌تر از نصف داده‌ها ۱ هستند و مُد برابر ۰ است. اما اگر تعداد ۱ها، از این عدد بزرگ‌تر بود، یعنی بیشتر از نصف داده‌ها ۱ هستند و مُد برابر ۱ است. پس تابع مُد بر روی داده‌های باینری، algebraic است.

- میانه: holistic

باید کل داده‌ها به صورت یکپارچه در نظر گرفته شوند و sort شوند، تا بتوان عنصر وسط را تشخیص داد. در نتیجه تابع میانه، holistic است.

- واریانس: algebraic

با توجه به فرمول بیان شده برای محاسبه‌ی واریانس، به کمک توابع average() (که algebraic است و به کمک توابع count() و sum() که distributive هستند، قابل محاسبه است) و sum() (که distributive است)، می‌توان واریانس را محاسبه کرد. پس، می‌توان تابع واریانس را به کمک یک‌سری توابع distributive محاسبه کرد. در نتیجه، واریانس یک تابع algebraic است.

همچنین، با توجه به اینکه، standard deviation یک تابع algebraic است و اگر آن را به توان ۲ برسانیم، واریانس بدست می‌آید، می‌توان نتیجه گرفت که واریانس نیز algebraic است.

سوال ۳:

الف) تعداد cuboid ها در یک data cube با N بعد، تعداد cuboid ها برابر است با 2^N. در این data cube با ۱۰ بعد، تعداد cuboid ها برابر است با:

$$2^{10} = 1024$$

ب) ۳ سلول بسته‌ی غیرتهی در این data cube وجود دارد:

$(a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10})$: 1

$(a_1, b_2, a_3, b_4, b_5, b_6, b_7, b_8, a_9, b_{10})$: 1

$(a_1, *, a_3, *, *, *, *, *, a_9, *)$: 2

ج) تعداد سلول‌های aggregate غیرتهی: 2037

$$2 \times 2^{10} - 2 - 2 - 7 = 2^{11} - 11 = 2048 - 11 = 2037$$

دو سلول پایه

دو سلول تهی

سلول‌های مشترک:

$(a_1, *, *, *, *, *, *, *, *, *)$: 2

$(*, *, a_3, *, *, *, *, *, *, *)$: 2

$(*, *, *, *, *, *, *, *, a_9, *)$: 2

$(a_1, *, a_3, *, *, *, *, *, *, *)$: 2

$(a_1, *, *, *, *, *, *, *, a_9, *)$: 2

$(*, *, a_3, *, *, *, *, *, a_9, *)$: 2

$(a_1, *, a_3, *, *, *, *, *, a_9, *)$: 2

د) ۱ سلول aggregate بسته‌ی غیرتهی در این data cube وجود دارد:

$(a_1, *, a_3, *, *, *, *, *, a_9, *)$: 2

ه) ۷ سلول aggregate غیرتهی با شرط Having count(*) ≥ 2 در iceberg cube متناظر وجود دارد:

تمام ancestor های $(a_1, *, a_3, *, *, *, *, *, a_9, *)$:2 و خودش؛ یعنی:

$(a_1, *, *, *, *, *, *, *, *, *)$: 2

$(*, *, a_3, *, *, *, *, *, *, *)$: 2

$(*, *, *, *, *, *, *, *, a_9, *)$: 2

$(a_1, *, a_3, *, *, *, *, *, *, *)$: 2

$(a_1, *, *, *, *, *, *, *, a_9, *)$: 2

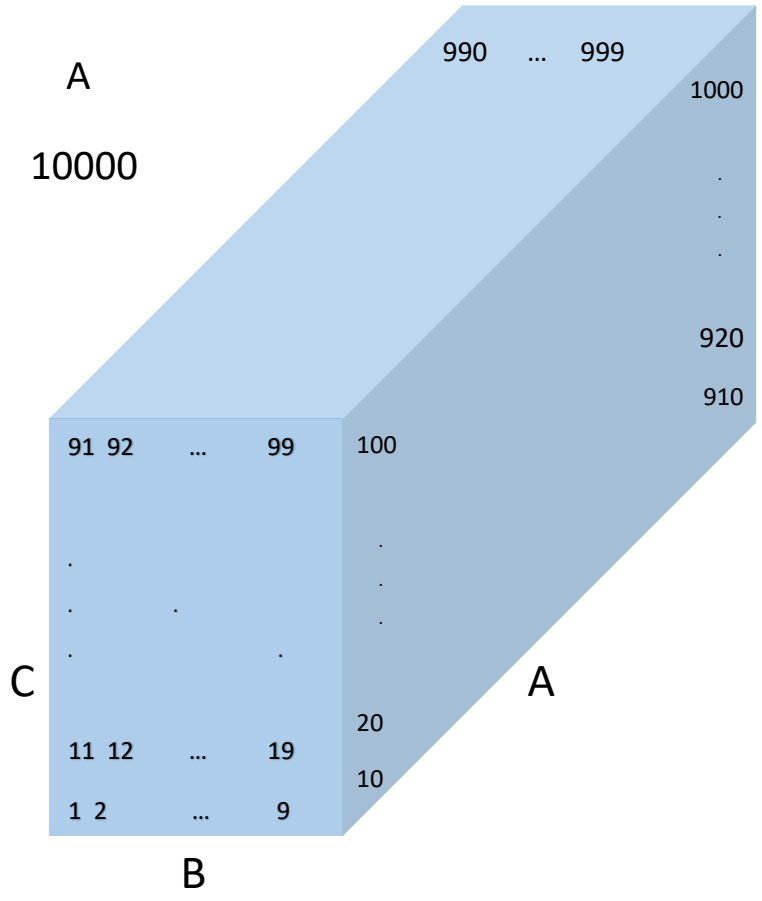
$(*, *, a_3, *, *, *, *, *, a_9, *)$: 2

$(a_1, *, a_3, *, *, *, *, *, a_9, *)$: 2

سوال ۴:

الف) هر یک از ابعاد cube به ۱۰ تکه (chunk) تقسیم شده اند. یعنی در کل، این cube از ۱۰۰۰ سلول به شکل زیر تشکیل شده است:

بعد :	B	C	A
کاردینالیتی :	100	1000	10000



- ترتیب بهینه‌ی پیمایش سلول‌های این cuboid، به صورت شماره گذاری شده در شکل بالاست.
- برای محاسبه‌ی هر سلول از cuboid دوبعدی CA، ۱۰ سلول از cuboid بالا باید پیمایش شود. برای مثال برای محاسبه‌ی C_0A_0، باید سلول‌های ۱ تا ۱۰ را پیمایش کرد. برای این کار به اندازه‌ی ۱ سلول فضا نیاز است. که نوشته می‌شود و آزاد می‌شود.
- برای محاسبه‌ی هر سلول از cuboid دوبعدی BA، حداکثر ۱۰۰ سلول از cuboid بالا باید پیمایش شود. برای این کار به اندازه‌ی ۱ سطر فضا نیاز است. که نوشته می‌شود و آزاد می‌شود.
- برای محاسبه‌ی هر سلول از cuboid دوبعدی BC، حداکثر کل سلول‌های cuboid بالا باید پیمایش شود. برای این کار به اندازه‌ی کل ۱۰۰۰ سلول فضا نیاز است. که نوشته می‌شود و آزاد می‌شود.
- اگر بتوان برای یک سلول از CA، یک سطر از BA و کل BC فضا اختصاص داد، می‌توان محاسبات را به صورت همزمان انجام داد. و لازم نیست بیش از یک بار این سلول‌ها را به حافظه‌ی اصلی آورد.
- در بخش بعد، میزان فضای لازم برای حافظه برای انجام این محاسبات، محاسبه شده‌است.

ب)

- تعداد سلول‌های قرار گیرنده در حافظه:

یک سلول از CA　　یک سطر از BA　　کل BC　　یک سلول

$$\left(\frac{1000}{10} \times \frac{10000}{10}\right) + \left(100 \times \frac{10000}{10}\right) + (100 \times 1000) + 1$$

$$= (100 \times 1000) + (100 \times 1000) + (100 \times 1000) + 1$$

$$= 3 \times 10^5 + 1 = 300001$$

- هر سلول، یک معیار را در ۴ بایت ذخیره می‌کند:

$$4 \times 300001 = 1.200.004\ Byte$$

- $1.200.004$ بایت فضا در حافظه مورد نیاز است، تا بتوان با یک دور پیمایش سلول‌های cuboid سه بعدی، cuboid های دوبعدی را ساخت.

ج)

- سایز cuboid های دوبعدی:

cuboid دوبعدی:	CA	BA	BC
سایز:	$10^3 \times 10^4 = \mathbf{10^7}$	$10^2 \times 10^4 = \mathbf{10^6}$	$10^2 \times 10^3 = \mathbf{10^5}$

- cuboid یک بعدی A از روی CA و BA قابل محاسبه است. با توجه به این‌که BA کوچک‌تر از CA است، از BA برای محاسبه‌ی A استفاده می‌شود.

- cuboid یک بعدی B از روی BA و BC قابل محاسبه است. با توجه به این‌که BC کوچک‌تر از BA است، از BC برای محاسبه‌ی B استفاده می‌شود.

- cuboid یک بعدی C از روی CA و BC قابل محاسبه است. با توجه به این‌که BC کوچک‌تر از CA است، از BC برای محاسبه‌ی C استفاده می‌شود.

سوال ۵:

الف) بهترین ترتیب برای پردازش ابعاد، بدین صورت است که ابتدا B، سپس C و در نهایت A را پردازش کنیم. دلیل انتخاب این ترتیب این است که بهتر است ابعاد را به ترتیب نزولی تعداد مقادیر مختلفی که دارند، مرتب کنیم و آن‌ها را به ترتیب از بُعد با بیشترین تعداد مقادیر مختلف تا بعد با کم‌ترین تعداد مقادیر مختلف، برای پردازش انتخاب کنیم. دلیل این معیار انتخاب این است که هرچه توزیع یک بُعد، یکنواخت‌تر باشد و تعداد مقادیر مختلف بیشتری داشته باشد، اگر پردازش را از آن شروع کنیم، احتمال این‌که بتوان pruning انجام داد بیشتر می‌شود.

ب) داده‌ها بدین شکل خلاصه می‌شوند:

<table>
<tr><th>B</th><th>C</th><th>A</th><th>count</th></tr>
<tr><td>b1</td><td rowspan="2">c1</td><td rowspan="3">a1</td><td>1</td></tr>
<tr><td>b2</td><td>1</td></tr>
<tr><td rowspan="2">b3</td><td>c2</td><td>1</td></tr>
<tr><td>c3</td><td rowspan="2">a2</td><td>1</td></tr>
<tr><td>b4</td><td>c2</td><td>1</td></tr>
</table>

(minimum support = 2)

:iceberg cube -

(b_3, *, *): 2 → B

(*, c_1, *): 2 → C

(*, c_2, *): 2 → C

(*, c_1, a_1): 2 → CA

(*, *, a_1): 3 → A

(*, *, a_2): 2 → A

بقیه prune شدند.