LOOKML-DEVELOPER$^{Q\&As}$

LookML Developer

Pass Google LOOKML-DEVELOPER Exam with 100% Guarantee

QUESTION 1

A LookML developer creates an Explore that joins two views. The base view has information about users' interactions with the support team. The joined view contains data about the users. The support team using this Explore feels overwhelmed by the amount of data this Explore shows them and decides to just look at open tickets.

What should the developer add to the Explore in the model to achieve these requirements?

A. A filtered measure

B. The hidden parameter

C. The sql_always_where parameter

D. A relationship definition

Correct Answer: D

QUESTION 2

A developer has a persistent derived table view called user_facts that contains aggregated data for each user. The developer needs to query the data from this table in another derived table view called user_region_facts.

Which strategy should the developer use to write the query for user_region_facts that will leverage the existing derived table?

A. Use ${user_facts.SQL_TABLE_NAME} to reference the user_facts derived table.

B. Copy the name of the database table in the scratch schema for the user_facts derived table.

C. Writhe the query form user_facts into a common table expression (WITH user_facts AS...).

D. Write a subquery in the FROM clause and alias with ${user_facts}.

Correct Answer: C

QUESTION 3

A developer is building an e-commerce Explore with the following datasets: orders and users. The business user needs to be able to answer questions about sellers and buyers within the same Explore. Each order in the orders table reports a buyer and seller ID. The users table has the detailed information about the individual buyer and seller.

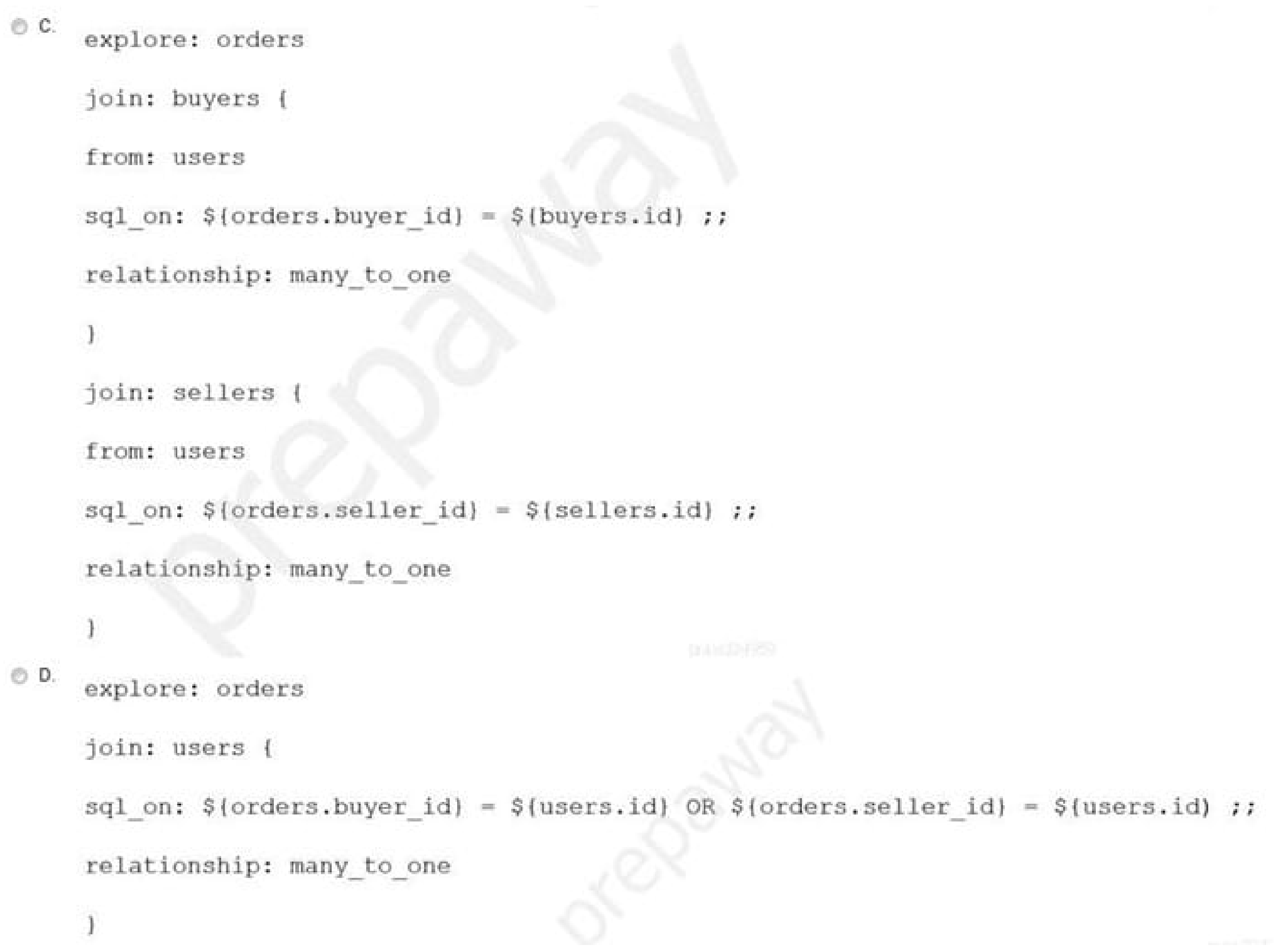
How should the Explore be defined to meet this requirement?

A.

```
explore: orders

join: buyers {

view_name: users

sql_on: ${orders.buyer_id} = ${buyers.id} ;;

relationship: many_to_one

}

join: sellers {

view_name: users

sql_on: ${orders.seller_id} = ${sellers.id} ;;

relationship: many_to_one

}
```

B.

```
explore: orders

join: users {

sql_on: ${orders.buyer_id} = ${users.id} AND ${orders.seller_id} = ${users.id} ;;

A relationship: many_to_one

}
```



C.

```
explore: orders

join: buyers {

from: users

sql_on: ${orders.buyer_id} = ${buyers.id} ;;

relationship: many_to_one

}

join: sellers {

from: users

sql_on: ${orders.seller_id} = ${sellers.id} ;;

relationship: many_to_one

}
```

D.

```
explore: orders

join: users {

sql_on: ${orders.buyer_id} = ${users.id} OR ${orders.seller_id} = ${users.id} ;;

relationship: many_to_one

}
```

A. Option A

B. Option B

C. Option C

D. Option D

Correct Answer: B

QUESTION 4

The developer is creating an Explore that includes the product users, and orders views that will meet the following guidelines.

Joins between the orders and users views should not incur high performance costs.

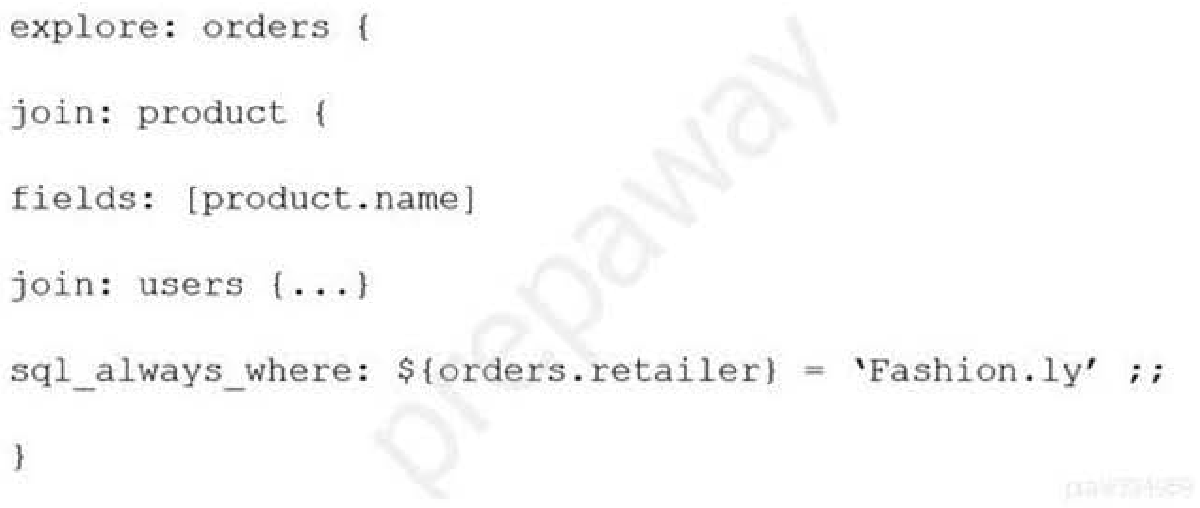
Users of this Explore will primarily be looking at data from the orders view.

Users of this Explore should only be able to see orders from the retailer “Fashion.ly”.

The only field the users need from the products view is product.name.

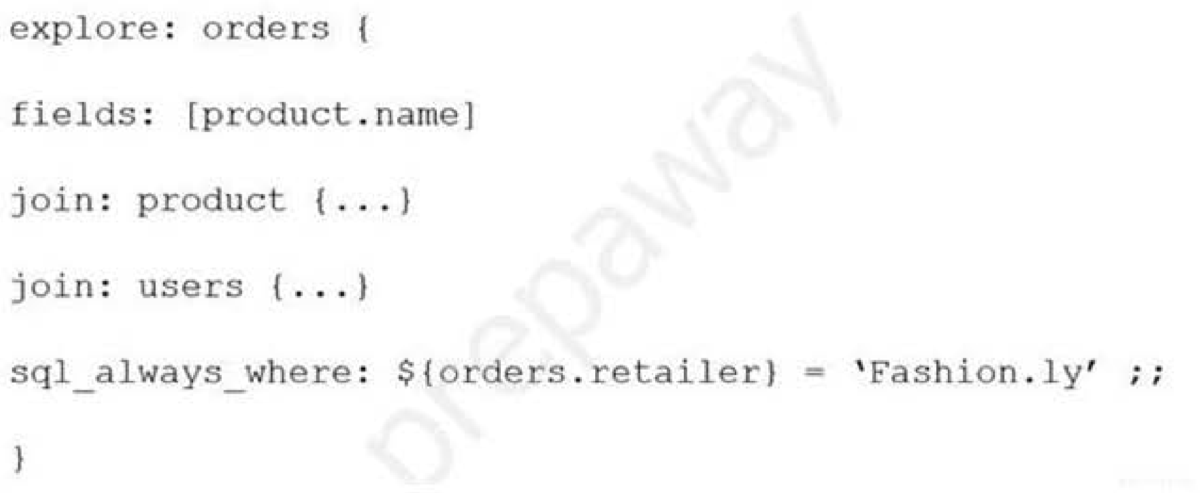
Which LookML should the developer use?

A.

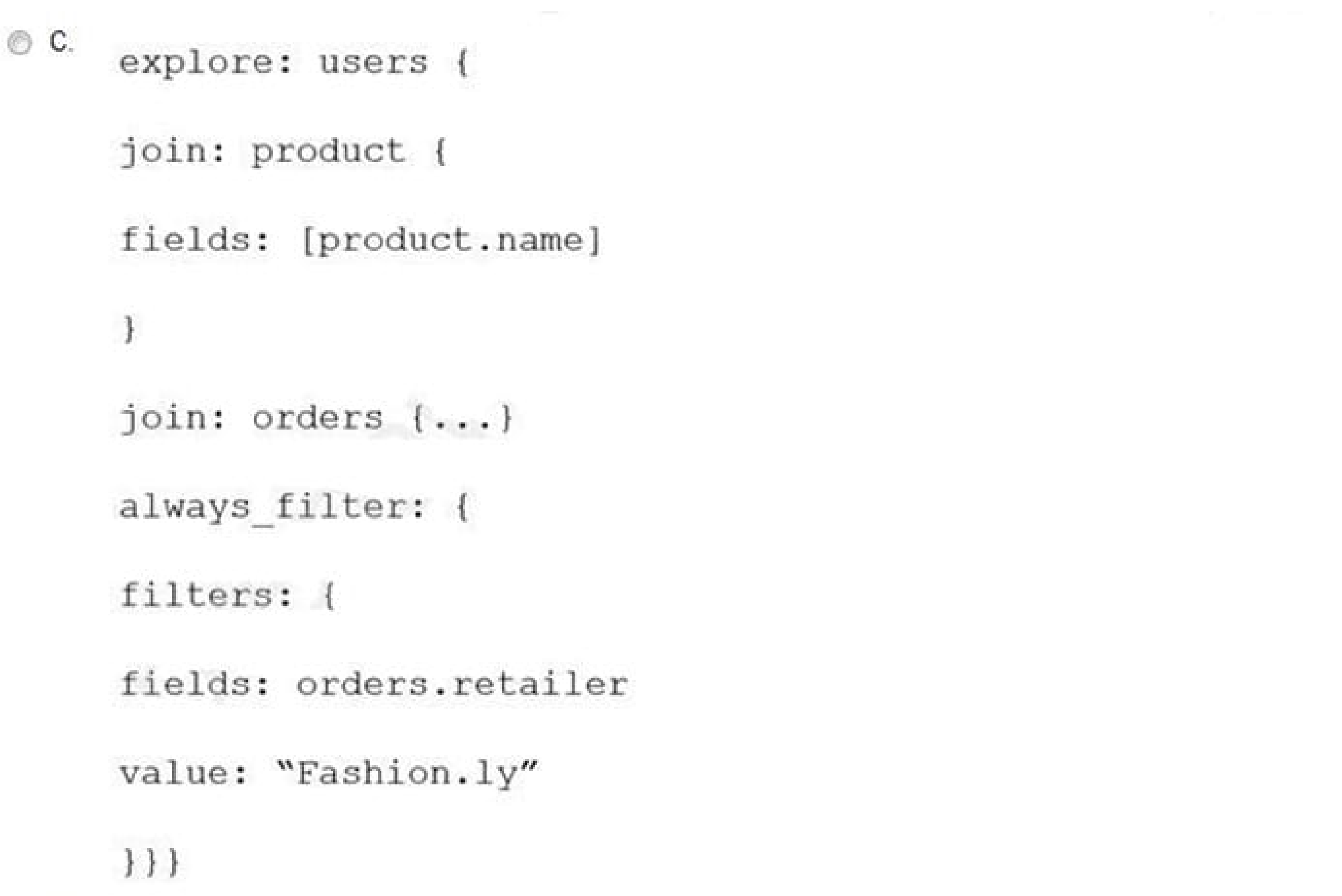


```
explore: orders {

join: product {

fields: [product.name]

join: users {...}

sql_always_where: ${orders.retailer} = 'Fashion.ly' ;;

}
```

B.

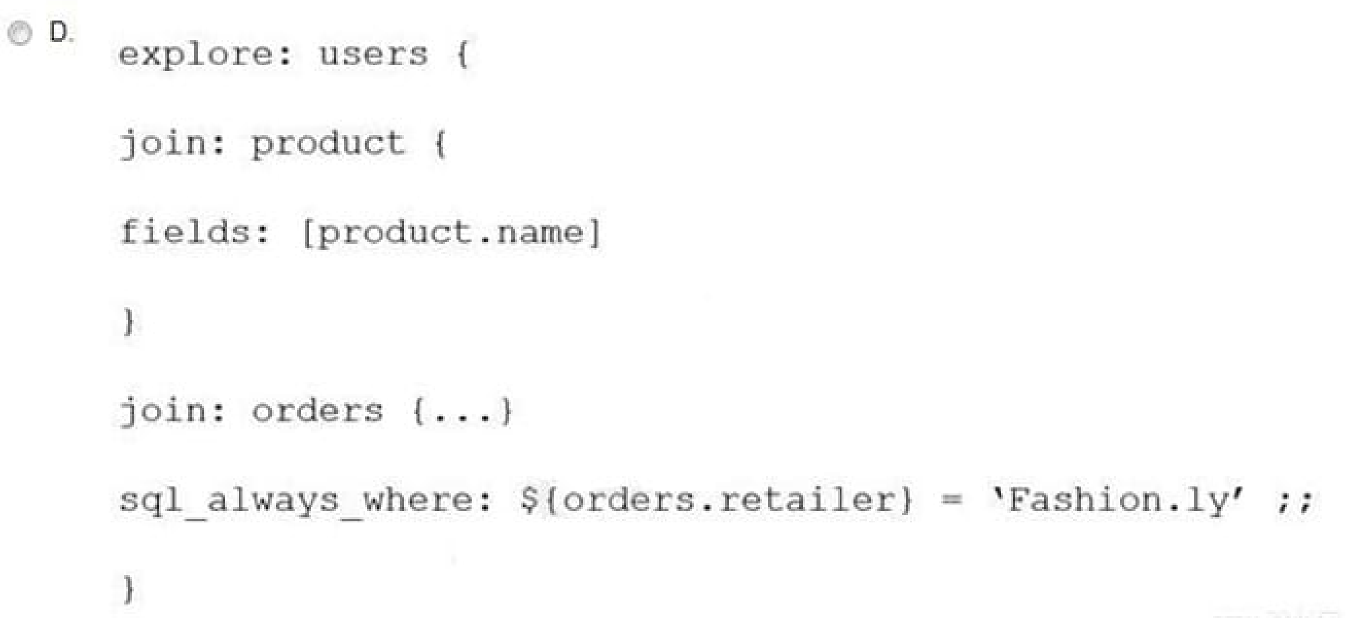


```
explore: orders {

fields: [product.name]

join: product {...}

join: users {...}

sql_always_where: ${orders.retailer} = 'Fashion.ly' ;;

}
```



C.

```
explore: users {

join: product {

fields: [product.name]

}

join: orders {...}

always_filter: {

filters: {

fields: orders.retailer

value: “Fashion.ly”

}}}
```



D.

```
explore: users {

join: product {

fields: [product.name]

}

join: orders {...}

sql_always_where: ${orders.retailer} = 'Fashion.ly' ;;

}
```

A. Option A

B. Option B

C. Option C

D. Option D

Correct Answer: D

QUESTION 5

A developer has User Specific Time Zones enabled for a Looker instance, but wants to ensure that queries run in Looker are as performant as they can be. The developer wants to add a datatype: date parameter to all dimension_group definitions without time data in a table-based view, so that time conversions don\\'t occur for these fields.

How can the developer determine to which fields this parameter should be applied through SQL Runner?

A. Open the Explore query in SQL Runner and validate whether removing the conversion from date fields changes the results.

B. Open the Explore query in SQL Runner to determine which fields are converted.

C. Use the CAST function in SQL Runner to ensure that all underlying fields are dates and conversions are not applied.

D. Use the Describe feature in SQL Runner to determine which fields include time data.

Correct Answer: C

QUESTION 6

A LookML developer builds a view that contains sensitive information. Only members of the Management group should have access to the view. The developer needs to restrict the view from appearing in the field picker for any Explore where it might be joined for users outside of the Management group.

Which LookML parameter should the developer use to meet this requirement?

A. access_grant

B. always_filter

C. access_filter

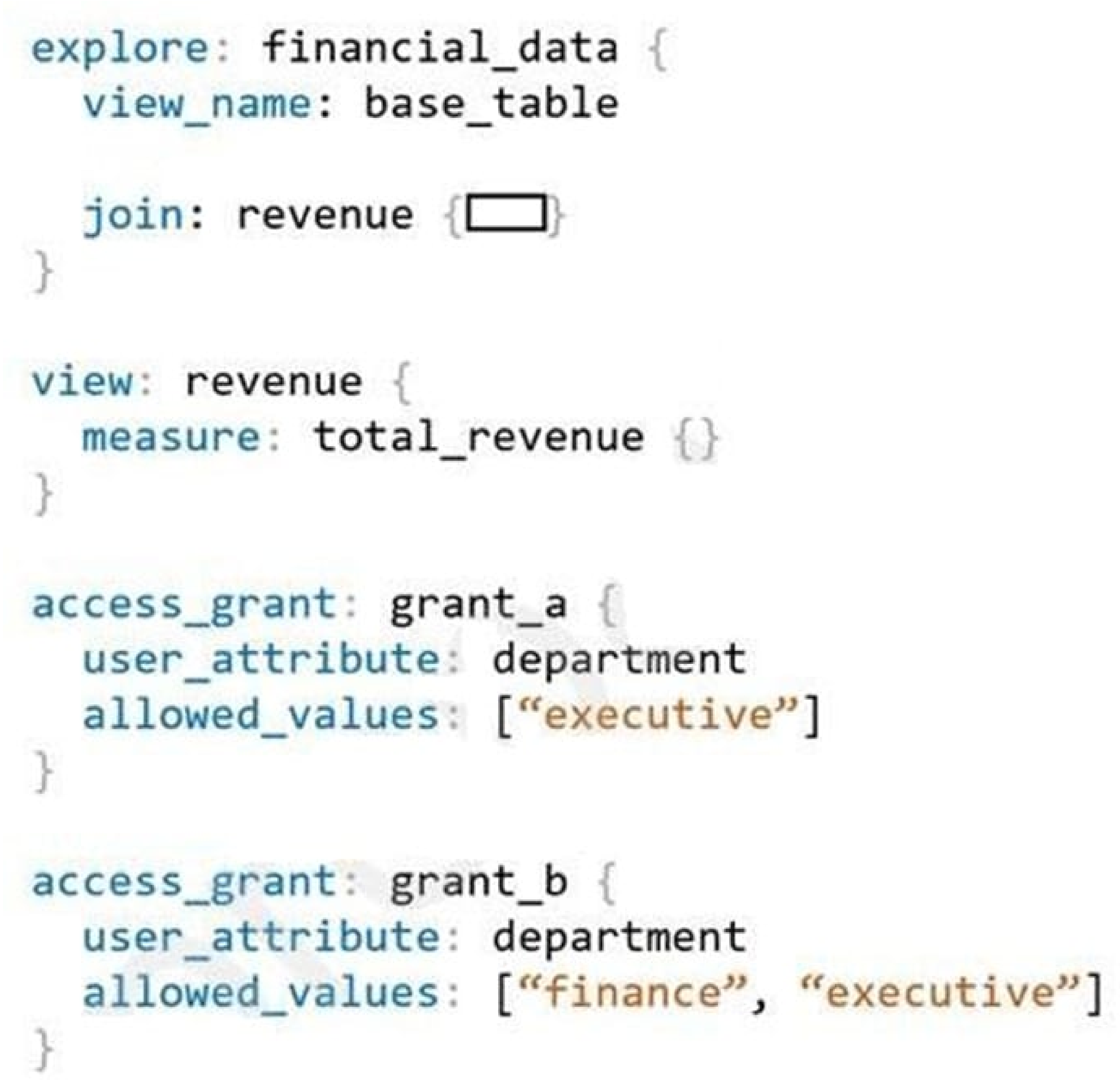
D. sql_always_where

Correct Answer: A

QUESTION 7

Only users with department attributes of Finance and Executive should be able to access the revenue view. Only users with the value of Executive for the department user attribute should be able to view the total_revenue field.

Given the code snippet below: How should the required access grants be structured to set up this system of access?



```
explore: financial_data {
  view_name: base_table

  join: revenue {    }
}

view: revenue {
  measure: total_revenue {}
}

access_grant: grant_a {
  user_attribute: department
  allowed_values: ["executive"]
}

access_grant: grant_b {
  user_attribute: department
  allowed_values: ["finance", "executive"]
}
```

A. required_access_grants: [grant_b] in the revenue view, required_access_grants: [grant_a] in the total_revenue field

B. required_access_grants: [grant_a] in the revenue view, required_access_grants: [grant_a, grant_b] in the total_revenue field

C. required_access_grants: [grant_b] in the financial_data Explore, required_access_grants: [grant_a] in the total_revenue field

D. required_access_grants: [grant_a, grant_b] in the revenue view, required_access_grants: [grant_a] in the total_revenue field

Correct Answer: B

QUESTION 8

A LookML developer finishes some LookML work and commits changes in their personal development branch. The developer is asked to Pull and Merge Other Changes.

What does this indicate?

A. A change has been deployed to a shared branch.

B. A change has been committed in another developer\\'s personal branch.

C. A change has been committed in another shared branch.

D. A change has been deployed to production.

Correct Answer: B

QUESTION 9

Users report that the main dashboard has been slow to show results.

Which two options should the developer evaluate to improve dashboard performance? (?hoose two.)

A. Number of databases used by dashboard elements

B. Number of queries used by the dashboard

C. Ratio of visualizations to text tiles D. Format used to deliver these reports

E. Amount of data rendered for each query

Correct Answer: BC

QUESTION 10

A developer wants to create a measure that shows the item count broken out by category. When a second,

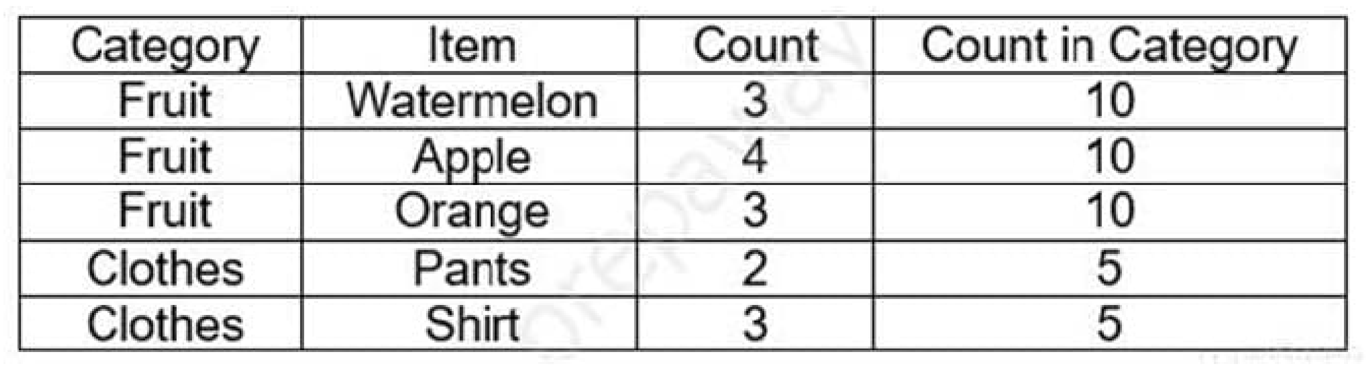
more granular dimension is added to the same query, the count broken out by category should still

represent the original aggregation and be duplicated on each line. The business wants this “count” in

“category” available in the Explore section without any additional work done by the end user. For example:

The Count column represents the count for each combination of Category and Item.

The Count in Category column represents the count for each Category only.



Category	Item	Count	Count in Category
Fruit	Watermelon	3	10
Fruit	Apple	4	10
Fruit	Orange	3	10
Clothes	Pants	2	5
Clothes	Shirt	3	5

How can the developer address this need with a LookML object?

A. Create a measure filtered on Category, and make the filter value controlled by a parameter.

B. Calculate the measure using a derived table, and then join that derived table back into the Explore.

C. Create a measure with type: sum_over_dimension, and make the dimension value controlled by a parameter.

D. Calculate the overall count using table calculations in the Explore.

Correct Answer: B