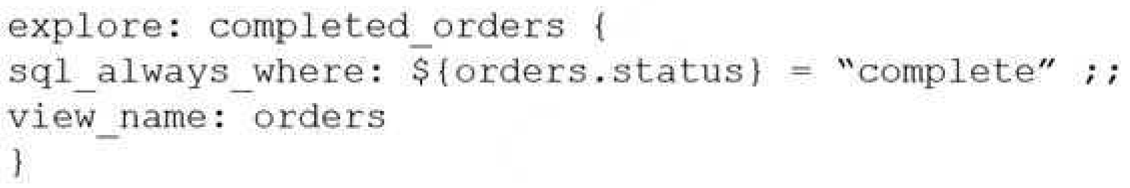
Latest Version: 6.0

Question: 1

A developer needs to add an Explore built off of the orders view, which surfaces only completed orders. An orders Explore exists that contains all order information. Fields from the orders view are also referenced in other existing views such as ${orders.fieldname}.

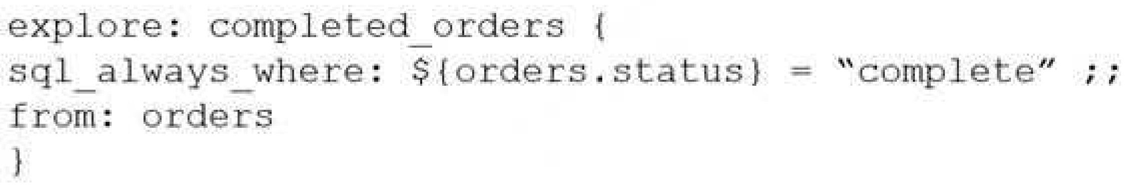
How should developer define a new Explore for completed orders and keep all field references working correctly?

A)



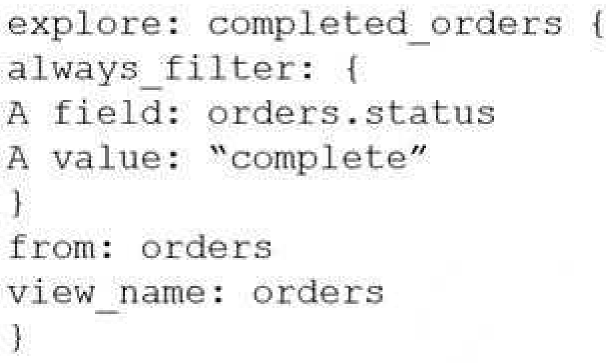
```
explore: completed_orders {
sql_always_where: ${orders.status} = “complete” ;;
view_name: orders
}
```

B)



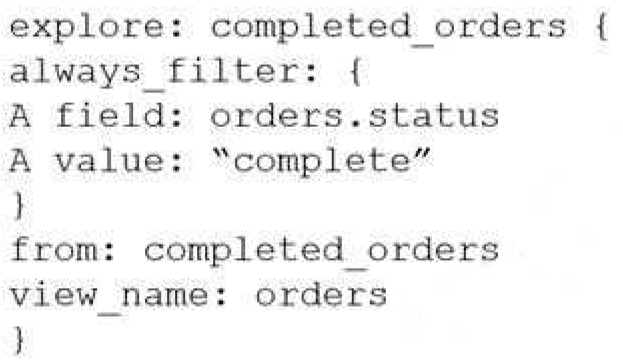
```
explore: completed_orders {
sql_always_where: ${orders.status} = “complete” ;;
from: orders
}
```

C)



```
explore: completed_orders {
always_filter: {
A field: orders.status
A value: “complete”
}
from: orders
view_name: orders
}
```

D)



```
explore: completed_orders {
always_filter: {
A field: orders.status
A value: “complete”
}
from: completed_orders
view_name: orders
}
```

A. Option A
B. Option B
C. Option C
D. Option D

Answer: C

Reference: https://hevodata.com/learn/understanding-looker-ml/

Question: 2

Business users report that they are unable to build useful queries because the list of fields in the Explore is too long to find what they need.
Which three LookML options should a developer use to curate the business user’s experience? (Choose three.)

A. Add a description parameter to each field with context so that users can search key terms.
B. Create a separate project for each business unit containing only the fields that the unit needs.
C. Add a group_label parameter to relevant fields to organize them into logical categories.
D. Use the hidden parameter to remove irrelevant fields from the Explore.
E. Use a derived table to show only the relevant fields.

Answer: A, C, E

Question: 3

A user reports that a query run against the orders Explore takes a long time to run. The query includes only fields from the users view. Data for both views is updated in real time. The developer runs the following query in SQL Runner and quickly receives results:
SELECT * FROM users.
What should the developer do to improve the performance of the query in the Explore?

A. Create an Explore with users as the base table.
B. Create a persistent derived table from the user’s query.
C. Create an ephemeral derived table from the user’s query.
D. Add persist_for: “24 hours” to the orders Explore.

Answer: A

Reference: https://docs.looker.com/data-modeling/learning-lookml/sql-runner

Question: 4

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.

Answer: C

Reference: https://community.looker.com/technical-tips-tricks-1021/how-looker-does-timezones-andhow-to-troubleshoot-them-25477

Question: 5

A LookML developer creates a new model and a test dashboard from the model. The developer shares the link to the new dashboard with users, but the users report that all they see is the “Model Not Found” error.
What is a possible cause of this issue?

A. The developer has not pushed the new model to Production Mode.
B. The developer has not added users to the new model set.
C. The users do not have permission to access this dashboard.
D. The new model is missing an Explore definition.

Answer: B