1st scenario of comparison – "Which class is more cohesive?"

DB_Backend vs. DB_InsertUpdate:

- **With orange background, explanation from the 4 responses which chose the second class (DB_InsertUpdate) as more cohesive**
- **With yellow background, explanation from the 30 responses which chose both classes having quite similar cohesion**
- **With blue background, explanation from the 45 responses which chose the first class (DB_Backend) as more cohesive**

13	Na primeira classe estão todas as funções necessárias para a conexão com o banco de dados, abertura e fechamento. Já no segundo código, além da inserção, ela também realiza operações de update. O ideal seria ter duas classes separadas.
15	A classe DB_Backend faz a conexão com o banco de dados. A classe DB_InsertUpdate de cara já tem duas responsabilidades (inserir e atualizar)
16	DB_Backend não faz referência para DB_InsertUpdate, enquanto o inverso é verdadeiro. DB_InsertUpdate depende de DB_Backend para implementar suas funcionalidades Exemplo de dependência em DB_InsertUpdate: DB_Backend.getConnection().createStatement();
17	Creio que as duas sejam bem específicas para as funções que realizam.
18	Para mim possuem o mesmo nível de coesão pois cada uma tem um único objetivo e seria complicado separar os métodos de cada classe.
19	Acredito que apesar da segunda classe implementar dois comportamentos (Incluir e Atualizar) esses comportamentos ao meu ver pertencem a um unico modulo.
20	A segunda classe que seria utilizada apenas para realizar insert e update também é utilizada para concatenar os valores do array em uma string. A primeira classe na minha análise (não conheço mysql) é utilizada para gerenciamento da conexão (abrir, fechar, verificar se está aberta a conexão, etc).
22	Embora o nome da classe DB_InsertUpdate indique uma composição de 2 operações (insert e update), e portanto, o que deve estar dentro dela é justamente operações para inserção e atualização, a coesão funcional parece negligenciada, pois inserção e atualização são 2 tarefas que endereçam problemas distintos. Eu diria que há uma coesão comunicacional, onde os dados de entrada/saída são compatíveis (os mesmos).
24	Classes com funções bem definidas.

25	Resposta curta e mais correta: feeling. Resposta longa e menos correta: A primeira classe tem uma responsabilidade bem definida, que é gerenciar conexões com o BD, e não precisa de informações das classes que farão uso dela. Já a segunda classe possui uma responsabilidade difícil de definir e faz suposições frágeis a respeito dos módulos que farão uso dela, como o modelo do BD.
26	The query builder should be in a different module/class and not inside the DB_InsertUpdate class.
28	Apesar dos nomes das classes não serem muito adequados, a classe DB_Backend tem uma única responsabilidade. A de manter um pool de conexões (erroneamente implementado, mas...). Já a classe DB_InsertUpdate em a responsabilidade de inserir e alterar toda e qualquer tabela no banco.. o que me parece uma responsabilidade muito ampla.. Caso você considere a responsabilidade da segunda seja exatamente esta. Alterar todas as tabelas do banco.. poderia-se pensar que ela é coesa.. Mas isto varia de pessoa para pessoa
29	DB_Backend seems to be more cohesive. It has some shared data(attributes) compared to DB_InsertUpdate. There are no attributes at all and all methods are static and work independently. Offtopic: I am not quite sure, but it seems to me that DB_InsertUpdate might be vulnerable against sql-injection.
30	A classe DB_Backend realiza operações apenas de comunicação com o banco de dados. Já a classe DB_InsertUpdate realiza duas operações, inserir e atualizar. Para a segunda classe ser mais coesa, ela teria que ser dividida em duas: DB_Insert e DB_Update
32	**A segunda classe tem métodos mais claros, legíveis e tem uma complexidade muito menor. No caso da primeira classe é possível subdividir os métodos tornando mais coesa a classe.**
33	Na primeira classe os métodos utilizam atributos internos. Isto não ocorre na segunda classe, que facilmente poderia ser decomposta em mais partes.
34	Verifiquei que as responsabilidades das duas classes estão bem definidas e nas mesmas possuem rotinas que apenas atendem ao que define a responsabilidade de cada uma.
35	A classe DB_Backend possui apenas a responsabilidade de abrir e fechar conexões. A segunda, apesar de genérica, está concentrando duas funcionalidade que não tem nada em comum: inserir e atualizar registros.
36	A primeira classe trata do gerenciamento das conexões. E a segunda da inserçaõ e atualziação dos dados utilizando a primeira classe.
37	A classe de backend não precisa de outras classes para gerenciar a conexão com banco. Já a classe InsertUpdate sempre precisará da classe Backend para inserir e atualizar dados no banco de dados. Logo existem funcionalidades de InsertUpdate que não estão contidas na mesma, ao contrário da classe de Backend. Por isso, a segunda é mais coesa.

38	Usei a definição de Steven et al., 1974: "em um módulo altamente coeso, todos os seus elementos estão relacionados para a realização de um único comportamento do software." Em minha opinião as duas classes tem alta coesão.
39	As duas estão designadas para mapear a mesma funcionalidade: persistência de dados
41	O meu raciocínio foi que as duas classes tem a mesma quantidade de responsabilidades, a classe DB_Backend gerencia conexões com o banco de dados, conectando e desconectando com o BD, enquanto a classe DB_InsertUpdade grava dados no banco de dados, inserindo ou atualizando informações.
42	a classe DB_Backend, é implementado apenas o que é destinado a classe, e a classe InsertUpdate recorre a classe DB_Backend para obter conexao para realizar suas operações, não implementando a conexao ao banco que seria necessaria em sua própria classe.
43	As duas classes possui a sua função específica e executa o que foi determinado.
44	If you analyse it carefully, and with some metrics, you could say that the Backend is more cohesive than InsertUpdate, as the second can be split into Insert and Update. However, this is not a problem if the sw is like this. There is a reason why INsert and Update were put together and it does not affect the maintenance of the sw in a negative way.
46	A primeira classe é responsável por uma unica tarefa, que é a conexão.
50	**A classe DB_Backend possui muitos métodos publicos que expôem seu funcionamento interno. Já a classe DB_InsertUpdate somente possui em sua interface os métodos especificos para o uso da mesma, e executa somente as tarefas que seu nome sugere.**
51	Levei em consideração o domínio que foi atribuído a cada classe e quais as ações que cada uma delas executava em seu código, assim avaliei se o comportamento de cada uma delas estava de acordo com o que lhe foi proposto.
54	O método DB_InsertUpdate.update é praticamente replicado e pode ser refatorado. O mesmo ocorre com o método DB_Backend.connect_DB. Por essa análise superficial, considerando que cada classe tem um método repetido, julgo terem o mesmo nível de coesão.
56	The first class provices a single type of service: database connection management. For this reason, it becomes difficult to split methods that open a new connection and close existing connections into different classes. The second class, in turn, is less cohesive as it aggregates services that could be split into different classes without creating redundant code across the classes.
57	A primeira só cuida da conexão com o banco. A segunda só faz as ações relacionadas a persistência, sem se preocupar com a conexão com o banco.

58	A primeira classe é mais coesa. Embora os métodos de conexão pudessem ser agregados em um só, eles executam uma mesma responsabilidade para a classe: conexão com o banco de dados. No caso da segunda classe, é possível dividi-la em duas novas: uma responsável pela inserção de novos registros no banco de dados e outra que cuide da atualização de registros já existentes.
59	Se olharmos ao propósito de casa uma das classes, então elas são igualmente coesas, pois o propósito de cada uma é bem claro e distinto. Se olharmos à estruturação das classes tendo em vista a reutilização de métodos dentro da própria classe, então a primeira classe, (DB_Backend) é mais coesa, pois na segunda classe (DB_InsertUpdate) existe bastante código repetido que poderia ser isolado em métodos reutilizáveis.
60	A classe de DB_BackEnd, tem uma responsabilidade única, ela manipula as conexões do banco, no entanto a DB_InsertUpdate é responsável por inserir e editar registros, sendo assim, ela possui menos coesão, já que possui mais responsabilidades.
61	DB_InsertUpdate realiza duas operações (insert e update). De acordo com o conceito de coesão da página anterior, a classe DB_Backend é mais coesa pois realiza apenas a conexão com o banco de dados.
62	Both classes execute only functions regarding their own responsibilities, i.e., the DB_Backend does only handles the database connection, while the DB_InsertUpdate does only insert and update data by using the connection provided by the DB_Backend class.
63	A classe DB_Backend trata apenas da conexão com o banco de dados, cria a conexão, fornece um método para retornar esta conexão, além de finalizar a conexão ao banco. A classe faz exatamente o que se propõe. A classe DB_InsertUpdate apesar de não ser muito elegante por utilizar substrings para compor clausuras SQL (minha opinião) é também tão coesa quando a DB_Backend, faz exatamente apenas o que se propõe, insere e atualiza dados no banco. Como fiz esta investigação? Primeiro verifico a assinatura e nomes dos métodos, tento identificar do que se trata e se estão no lugar onde deveriam, após verifico o conteúdo do método tentando identificar se o nome está de acordo com a implementação.
64	According to the definitions in the previous section, I would say that the DB management class is more cohesive because it basically handles a single responsibility. In the Insert/Update class, we could split the class in two, as the elements are related to the performance of a different behaviors of the software.
65	Segundo a definição anterior, uma classe muito coesa possui uma única responsabilidade. O nome da segunda classe já sugere que ela tem duas responsabilidades (inserir e fazer update). Na primeira classe, a responsabilidade parece ser uma só: gerenciar conexões. Apesar de que poderia ser vista como três responsabilidades: abrir,fechar, e pegar conexão. Mas todos os métodos compartilham variáveis em comum, parecem estar bem relacionados, então eu diria que é só uma responsabilidade.
66	Porque a segunda classe talvez acumule mais responsabilidades do que deveria

67	A segunda classe trata de conexão efetiva com banco e controle de conexões, logo assumindo duas responsabilidades que poderiam ser definidas em classes diferentes.
69	Eu avaliei os métodos públicos providos pelas classes e o quão eles estão relacionados com a responsabilidade de cada classe, a qual inferi pelo comentário descrevendo cada classe. Desta forma, cheguei a conclusão que em ambos os casos os métodos públicos estão relacionados com a responsabilidade de cada classe. Além disso, usei como auxílio a definição anterior de que uma classe altamente coesa é indivisível.
70	The first class joins all responsibilities regarding a databased connection. On the other hand, the second class brings information that could be split into more than a single class, e.g., a class handling inserts and another handling updates.
71	A opção foi escolhida principalmente porque a segunda classe contém várias conexões com a classe que realiza o backend. Tais ações são relativas a uma funcionalidade que é necessária para o funcionamento da classe DB_InsertUpdate, mas que não faz parte da funcionalidade da classe.
72	de acordo com a definição dada, a classe DB_InsertUpdate pode ser dividida em outras classes mas a Backend teria mais dificuldade em ser dividida
73	A primeira classe tem como única responsabilidade a conexão com o banco de dados. A segunda classe, possui como responsabilidade insert e update e é mais fácil de dividir em duas classes diferentes. Como foi mostrado na página anterior, Bieman e Kang (1995) declarou que uma classe altamente coesa deve ser difícil de dividir em duas ou mais partes. Tomando isso como base, a primeira parece ser mais coesa do que a segunda classe.
74	The fields from both classes are similiar regarding the behavior of each class.
75	A primeira classe é uma classe utilitária para conectar e desconectar do banco. A segunda classe poderia ser dividida em duas, uma para inserções outra para atualizações.
77	Levando com consideração a definição mostrada no passo anterior ("uma classe coesa se mostra difícil de ser fragmentada em outras classes"), eu vejo a DB_Backend como mais coesa que a DB_InsertUpdate, apesar de a classe de conexão ser maior, e portanto correr maior risco de lidar com diferentes responsabilidades. Essa ideia foi derrubada através de uma análise do código, quando cheguei à conclusão de que os métodos de conexão ao BD seria muito mais dificilmente fragmentados em diferentes classes do que os métodos de Inserção e Update de registros, presentes na classe DB_InsertUpdate.
79	As classes tem o mesmo nível de coesão por que fazem o que a ti é responsável, também a implementação da InsertUpdate não seja a melhor, mas a sua classe tem a responsabilidade que a faz executar os processos.
83	Não sei o suficiente sobre coesão para conseguir perceber qual tem o maior nível de coesão :(

84	Levei em consideração o objetivo, responsabilidade, da classe. A primeira classe tem o único objetivo de gerenciar a conexão.
86	**A principio, a classe DB_Backend possui duas responsabilidade: gerência do pool e estabelecimento/encerramento das conexões. Dessa forma, reutilizar a gerência de pool para outras finalidades ficaria prejudicado.**
88	Porquê DB_InsertUpdate depende de DB_Backend, mas DB_Backend não depende de DB_InsertUpdate. Assim, DB_InsertUpdate deve estar sempre acoplada a DB_Backend, o que diminui a coesão.
89	Dado o objetivo que é dado a cada classe acredito que as duas foram coesas com o que pretendiam realizar. A primeira trata da conexão com o BD e nada mais e a segunda trata da inserção e alteração de dados no BD sendo assim não fazem mais ou além dos objetivos que lhe foram atribuídos.
90	A linha de comando Class.forName("com.mysql.jdbc.Driver") torna a classe menos flexível. É como se a classe estivesse presa ao MySQl.
91	Se analisarmos a coesão sob a perspectiva de que ela é medida de acordo com o relacionamento dos elementos internos, a classe DB_Backend é uma classe mais coesa do que a classe DB_InsertUpdate. Além disso, é mais difícil conseguir decompor esta classe.
93	Backend is harder to split. Both operations Connect and Close are quite bonded to itself, since the DB handle is kept in the class (public static Connection). The second class, in theory, could be divided into a Inserter and a Updater classes with no harm.
94	A primeira classe deixa claro que seu objetivo é gerenciar a conexão com o banco de dados. Internamente, de fato, seus métodos não fazem nada a mais que isso. A segunda classe possui apenas métodos para inserir e atualizar registros no banco de dados (BD), que é o objetivo da classe. Contudo, de um lado, e segundo uma das referências anteriores, essa classe poderia ser dividida em duas (DB_Insert e DB_Update) e, por tanto, não estaria tão coesa. Por outro lado, o autor da classe deixou claro que o objetivo da classe é "escrever" no BD e, nesse caso, tanto insert quanto update estão dentro desse objetivo. Dessa forma, eu a considera tão coesa quanto a primeira classe.
95	A primeira classe não corresponde a sua descrição, pois não manipula a conexão com o banco de dados e sim o pool de conexões. Os atributos e métodos não são completamente coesos, pois misturam a manipulação de uma unica conexão, como login, senha, endereço da base etc, com métodos que dizem respeito a manipulação/criação do conjunto de conexões. Já a segunda classe mistura características de operações na base de dados e query builder, dessa forma caracterizando a falta de coesão pelo excesso de responsabilidade.
97	The DB_Backend class only contains methods to handle "connect to DB" feature. The DB_InsertUpdate class also handle insert/update the data base so both classes are cohesive. They implement specific functionalists without using other classes. They also focus on specific functionalities.

98	The members of DB_InsertUpdate are quite unrelated; for DB_Backend, the methods are concerned with fields of the same class.
99	Both classes do what you described they would, and nothing else. Both classes log exception, which is crosscutting, but since both class do that, I feel they have similiar cohesion.
102	* Class DB_Backend has 3 jobs: connect to the db, disconnect to the db, and track the connection and the connection information. * DB_InsertUpdate is not actually an OO class, it is a set of functions in a module. If java had modules rather than classes we would put these methods into a module as functions. DB Insert Update does 2 things. Insert is actually quite different than update, although both are relevant to writing.
105	I consider the first class more cohesive. It only deals with database connections. The second class deals with two types of db operations: insert, and update. In this example, the second class might still be considered a cohesive class, but less cohesive relative to the first.
106	It is quite difficult to assess the cohesion of a class by itself and with as little context as is provided in this question. That being said, none of the two classes are clearly non-cohesive (both have a clear purpose and the functionalities included in each are indeed conceptually related). Beyond that, very little can be said.
107	The responsibility for the DB_Backend class was clear to me; as the name of the class implies, it keeps and manages the lifecycle of some connections to a database in the backend. On the other hand, the second class seems to have no responsibility; it has no attribute and only contains some static methods. This class is just a placeholder for these utility methods (or functions, since they have no side effects to the class). The class might never be instantiated.
110	Ambos estão resolvendo um problema específico. Mas existe problemas graves de: - repetição de código - DB_InserUpdate depende de DB_Backend (Deveria usar a Injeção de Dependência e Segregação de Interface) - DB_Backend depende do MySQL (Deveria usar a Injeção de Dependência e Segregação de Interface)
113	In my opinion the DB_Backend is a bit more cohesive than DB_InsertUpdate but not by much. Both classes are very easy to understand. However, DB_Backend seems a bit more cohesive because all the fields (conns, con_pool, Host, etc.) and all the methods (DB_Backend, init, connect_DB, etc.) implementing this database backend functionality are inside the class. On the other hand, the DB_InsertUpdate class does not have everything “encapsulated” in that class, and is also implementing 2 related (but slightly different) concepts: insert and update.
114	Each class is specialize to a behavior (connexion vs. update)
115	DB_Backend class encapsulates data for a database connection and provides methods for the connection. DB_InsertUpdate class provides only functions. The static methods could be integrated in other DB access class, e.g. DB_Backend.

116	A primeira classe lida, em sua maior parte, com atributos definidos na própria classe. Além disso, é possível perceber claramente que a classe representa um conceito (gerenciamento de conexões com BD). Isso ajuda na leitura e entendimento da classe. Por outro lado, a segunda classe já possui de cara um indício de que não é coesa: não possui nem um atributo e possui métodos grandes que utilizam métodos de várias outras classes.
117	Verifiquei se as classes fazem o que foram criadas para fazer, sem extender suas responsabilidades, e na minha opinião ambas são coesas.
118	Because the methods belonging to each of the two classes seems to be very particular to the intention described below
119	Based on the class name and its method names, both classes look like their methods should be included in those classes and no in any other class.
122	Although the design of the first class might not be considered optimal, it at least provides higher cohesion than the second. That is because the private fields of the class are used for database connectivity from all the methods. And the responsibility of the class is driver-level database connectivity. However, it should be noted that it mixes in Messaging, JOptionPane and Logging calls. These are cross-cutting concerns. At the very least, the class should use one single method of providing messages or errors, and not a mixed bunch of different calls to different classes. That responsibility is better suited to be in the caller of the class. The second class, on the other hand, provides only static methods that do not share any code or fields and also contain similar/duplicate lines. They also represent two responsibilities: Insert and Update. It can therefore be split easily into two classes.
123	Porque a segunda classe DB_InsertUpdate precisa da primeira DB_Backend para funcionar
124	Todos os métodos e propriedades das classes dizem respeito ao problema que elas foram destinadas a resolver. Apesar de achar estranho uma modelagem que dedica uma classe apenas para gerenciar conexões e outra apenas para cuidar de inserções e atualizações no banco de dados, não me parece que a coesão das mesmas foi afetada.
125	**a primeira se propõe a manusear a conexão com o banco de dados estando inclusas aí as atividades de abertura, fechamento e recuperação de uma conexão. já a segunda somente atualiza o estado do banco de dados.**