

ORACLE

Oracle MySQL

HeatWave LakeHouse WorkShop – TDC Brasil 2023

Ana Paula Sales

MySQL Solution Engineer

Ana.sales@oracle.com

Narciso Junior

MySQL Solution Engineer

Narciso.junior@oracle.com



MySQL HeatWave + LakeHouse – O que são?

- O **MySQL HeatWave** é um serviço de banco de dados totalmente gerenciado, alimentado pelo acelerador de consulta em memória do HeatWave. É o único serviço de nuvem que combina transações, análises em tempo real em data warehouses e data lakes e machine learning em um banco de dados MySQL, sem a complexidade, a latência, os riscos e os custos da duplicação de ETL.
- Com o **MySQL HeatWave Lakehouse**, os clientes podem consultar **meio petabyte** de dados no armazenamento de objetos e aproveitar todos os benefícios do HeatWave, mesmo quando seus dados são armazenados **fora** de um banco de dados MySQL. Com o **HeatWave AutoML**, desenvolvedores e analistas de dados podem criar, treinar, implementar e explicar modelos de machine learning no MySQL HeatWave sem mover dados para um serviço de ML separado.
- Vamos ver na prática como eles funcionam!



Getting Started: Acessando o Cloud Shell e utilizando MySQL Shell

- **Pré requisitos:**

- Para essa demonstração, será utilizado o Cloud Shell. O Cloud Shell é uma máquina virtual pequena que executa um shell bash e é acessada por meio do Console do Oracle Cloud (OCI), na página inicial.
- Para essa etapa, é necessário que você tenha uma conta Oracle trial ou paga.

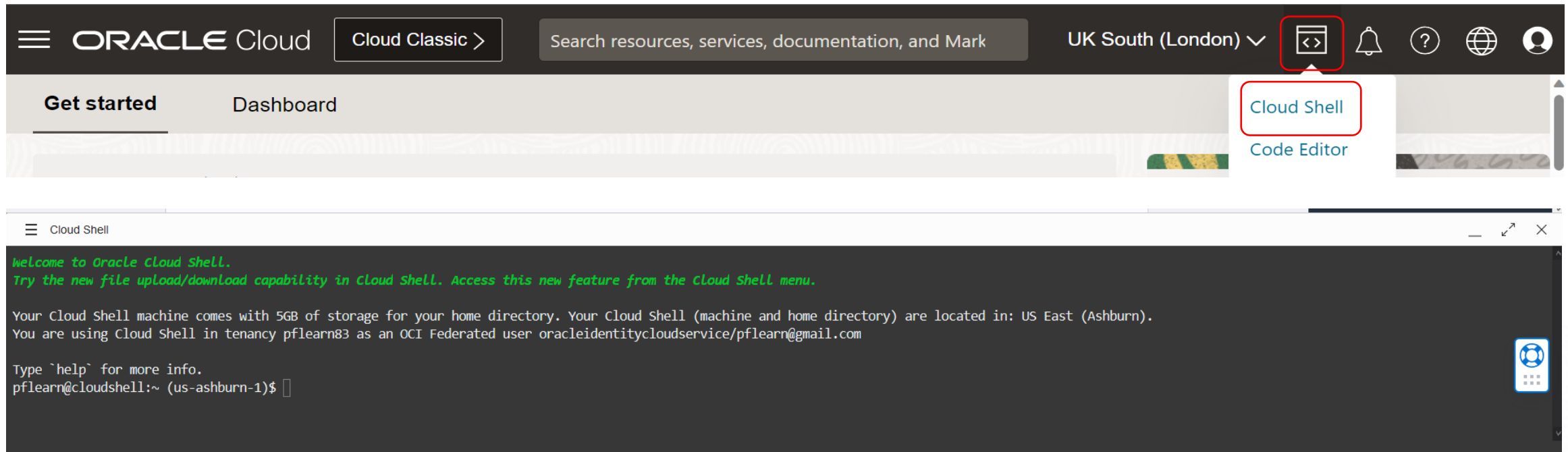
Link para acessar a OCI:

<https://cloud.oracle.com/>



Task 1: Acessando o Cloud Shell

- Para iniciar o Cloud Shell do Oracle, vá para a pagina inicial da OCI e clique no ícone do Cloud Shell no canto superior direito da página. Isso abrirá o Cloud Shell no navegador:



Task 2: Conectando na Base de Dados utilizando MySQL Shell

- O CloudShell já possui o MySQL Shell instalado, então, é necessário apenas autenticar-se e seguir com os próximos passos.
- Para seguir com a conexão, execute os comandos abaixo:

```
mysqlsh -utdc -p -h 132.145.23.69 --sql
```

- Para a senha, insira: @Tdc2023
- Para a pergunta sobre salvar a senha, aperte Enter ou digite “N”.

```
ana_sales@cloudshell:~ (uk-london-1)$ mysqlsh -utdc -p -h 132.145.23.69 --sql
Please provide the password for 'tdc@132.145.23.69': *****
Save password for 'tdc@132.145.23.69'? [Y]es/[N]o/Ne[v]er (default No):
```


Task 2: Conectando na Base de Dados utilizando MySQL Shell

- Exemplo de output esperado:

```
MySQL Shell 8.0.34-commercial

Copyright (c) 2016, 2023, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
Creating a session to 'tdc@132.145.23.69'
Fetching global names for auto-completion... Press ^C to stop.
Your MySQL connection id is 1458
Server version: 8.0.34-u3-cloud MySQL Enterprise - Cloud
No default schema selected; type \use <schema> to set one.
MySQL 132.145.23.69:3306 ssl SQL >
```

Task 2: Conectando na Base de Dados utilizando MySQL Shell

- Para visualizar as bases já carregadas na instância, execute no Cloud Shell o comando abaixo:

show databases;

```
MySQL 132.145.23.69:3306 ssl SQL > show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql_customer_orders |
| performance_schema |
+-----+
3 rows in set (0.0866 sec)
MySQL 132.145.23.69:3306 ssl SQL >
```

Task 2: Conectando na Base de Dados utilizando MySQL Shell

- Para visualizar o total de linha em cada uma das tabelas da base “mysql_customer_orders”:

```
SELECT table_name, table_rows FROM INFORMATION_SCHEMA.TABLES WHERE  
TABLE_SCHEMA = 'mysql_customer_orders';
```

```
MySQL 132.145.23.69:3306 ssl SQL > SELECT table_name, table_rows FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA = 'mysql_customer_orders';
```

TABLE_NAME	TABLE_ROWS
customer_order_products	NULL
customers	19454
delivery_orders	0
order_items	2559163
orders	34539902
product_orders	NULL
products	71
store_orders	NULL
stores	23
users	2

```
10 rows in set (0.0803 sec)
```


Task 3: Executando Queries com o HeatWave

- Pelo output do comando anterior, pode-se observar que a base my_sql_customer_orders é uma base de dados com tabelas que possuem milhões de linhas.

```
MySQL 132.145.23.69:3306 ssl SQL > SELECT table_name, table_rows FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA = 'mysql_customer_orders';
```

TABLE_NAME	TABLE_ROWS
customer_order_products	NULL
customers	19454
delivery_orders	0
order_items	2559163
orders	34539902
product_orders	NULL
products	71

- Para utilizar a base mysql_customer_orders, execute o comando abaixo:

USE mysql_customer_orders;

Task 3: Executando Queries com o HeatWave

- Como visto anteriormente, o motor HeatWave pode acelerar suas consultas complexas e workloads analíticos. Agora, você poderá ver isso na prática! Siga a instrução abaixo:
- Execute no Cloud Shell a query abaixo, que é uma lista das compras totais dos clientes no ano, por mês:

```
select `o`.`ORDER_ID` AS `order_id`, `o`.`ORDER_DATETIME` AS `ORDER_DATETIME`,  
  `o`.`ORDER_STATUS` AS `order_status`, `c`.`CUSTOMER_ID` AS `customer_id`,  
  `c`.`EMAIL_ADDRESS` AS `email_address`, `c`.`FULL_NAME` AS `full_name`,  
  sum(`oi`.`QUANTITY` * `oi`.`UNIT_PRICE`) AS `order_total`,  
  `p`.`PRODUCT_NAME` AS `product_name`, `oi`.`LINE_ITEM_ID` AS `LINE_ITEM_ID`,  
  `oi`.`QUANTITY` AS `QUANTITY`, `oi`.`UNIT_PRICE` AS `UNIT_PRICE`  
from (((`orders` `o` join `order_items` `oi` on((`o`.`ORDER_ID` = `oi`.`ORDER_ID`)))  
  join `customers` `c` on((`o`.`CUSTOMER_ID` = `c`.`CUSTOMER_ID`)))  
  join `products` `p` on((`oi`.`PRODUCT_ID` = `p`.`PRODUCT_ID`)))  
group by `o`.`ORDER_ID`, `o`.`ORDER_DATETIME`, `o`.`ORDER_STATUS`, `c`.`CUSTOMER_ID`,  
  `c`.`EMAIL_ADDRESS`, `c`.`FULL_NAME`, `p`.`PRODUCT_NAME`,  
  `oi`.`LINE_ITEM_ID`, `oi`.`QUANTITY`, `oi`.`UNIT_PRICE` limit 10;
```

Task 3: Executando Queries com o HeatWave

- Utilizando o HeatWave Cluster, o tempo de execução da query foi de aproximadamente 0.44 segundos:

```
Cloud Shell
MySQL 132.145.23.69:3306 ssl mysql_customer_orders SQL > select `o`.`ORDER_ID` AS `order_id`,`o`.`ORDER_DATETIME` AS `ORDER_DATETIME`,
-> `o`.`ORDER_STATUS` AS `order_status`,`c`.`CUSTOMER_ID` AS `customer_id`,
-> `c`.`EMAIL_ADDRESS` AS `email_address`,`c`.`FULL_NAME` AS `full_name`,
-> sum((`oi`.`QUANTITY` * `oi`.`UNIT_PRICE`)) AS `order_total`,
-> `p`.`PRODUCT_NAME` AS `product_name`,`oi`.`LINE_ITEM_ID` AS `LINE_ITEM_ID`,
-> `oi`.`QUANTITY` AS `QUANTITY`,`oi`.`UNIT_PRICE` AS `UNIT_PRICE`
-> from (((`orders` `o` join `order_items` `oi` on((`o`.`ORDER_ID` = `oi`.`ORDER_ID`)))
-> join `customers` `c` on((`o`.`CUSTOMER_ID` = `c`.`CUSTOMER_ID`)))
-> join `products` `p` on((`oi`.`PRODUCT_ID` = `p`.`PRODUCT_ID`)))
-> group by `o`.`ORDER_ID`,`o`.`ORDER_DATETIME`,`o`.`ORDER_STATUS`,`c`.`CUSTOMER_ID`
-> ,`c`.`EMAIL_ADDRESS`,`c`.`FULL_NAME`,`p`.`PRODUCT_NAME`
-> ,`oi`.`LINE_ITEM_ID`,`oi`.`QUANTITY`,`oi`.`UNIT_PRICE` limit 10;
```

order_id	ORDER_DATETIME	order_status	customer_id	email_address	full_name	order_total	product_name	LINE_ITEM_ID	QUANTITY	UNIT_PRICE
33480702	2022-02-10 00:00:00	COMPLETE	70	santiago.vautus@internalmail	Santiago Vautus	156.64	Girl's Trousers (Red)	1	4	39.16
6718553	2022-10-31 00:00:00	COMPLETE	57	michael.smith@internalmail	Michael Smith	159.12	Girl's Pyjamas (White)	1	4	39.78
133897138	2022-10-17 00:00:00	COMPLETE	83	salome.guisti@internalmail	Salome Guisti	241.95	Men's Pyjamas (Blue)	1	5	48.39
33463337	2022-04-06 00:00:00	CANCELLED	65	domingo.morano@internalmail	Domingo Morano	24.46	Women's Sweater (Brown)	2	1	24.46
937232137	2022-02-24 00:00:00	COMPLETE	90	ward.stepney@internalmail	Ward Stepney	74	Women's Jacket (Blue)	2	2	37.00
1476094702	2022-11-08 00:00:00	COMPLETE	94	joleen.himmelmann@internalmail	Joleen Himmelmann	39.89	Women's Socks (Grey)	1	1	39.89
669447936	2022-05-24 00:00:00	COMPLETE	88	jeraldine.audet@internalmail	Jeraldine Audet	126.72	Women's Coat (Black)	1	4	31.68
1472770000	2022-01-14 00:00:00	REFUNDED	94	joleen.himmelmann@internalmail	Joleen Himmelmann	185	Women's Jacket (Blue)	2	5	37.00
267802096	2022-08-27 00:00:00	COMPLETE	84	lovie.ritacco@internalmail	Lovie Ritacco	145.17000000000002	Men's Pyjamas (Blue)	3	3	48.39
535558536	2022-05-09 00:00:00	COMPLETE	87	carlotta.achenbach@internalmail	Carlotta Achenbach	148	Women's Jacket (Blue)	2	4	37.00

10 rows in set (0.4374 sec)



Task 3: Executando Queries com o HeatWave

- Para comparar a performance da execução da query com e sem o motor HeatWave ativado, desabilite a variável **use_secondary_engine** e execute novamente a query.
- Execute no Cloud Shell o comando abaixo:

```
SET SESSION use_secondary_engine=OFF;
```

```
10 rows in set (0.4574 sec)
MySQL 132.145.23.69:3306 ssl mysql_customer_orders SQL > SET SESSION use_secondary_engine=OFF;
Query OK, 0 rows affected (0.0780 sec)
MySQL 132.145.23.69:3306 ssl mysql_customer_orders SQL >
```

Task 3: Executando Queries com o HeatWave

- Agora que o motor foi desabilitado, execute novamente a query.
- Execute no Cloud Shell o comando abaixo:

```
select `o`.`ORDER_ID` AS `order_id`, `o`.`ORDER_DATETIME` AS `ORDER_DATETIME`,  
  `o`.`ORDER_STATUS` AS `order_status`, `c`.`CUSTOMER_ID` AS `customer_id`,  
  `c`.`EMAIL_ADDRESS` AS `email_address`, `c`.`FULL_NAME` AS `full_name`,  
  sum(`oi`.`QUANTITY` * `oi`.`UNIT_PRICE`) AS `order_total`,  
  `p`.`PRODUCT_NAME` AS `product_name`, `oi`.`LINE_ITEM_ID` AS `LINE_ITEM_ID`,  
  `oi`.`QUANTITY` AS `QUANTITY`, `oi`.`UNIT_PRICE` AS `UNIT_PRICE`  
from (((`orders` `o` join `order_items` `oi` on((`o`.`ORDER_ID` = `oi`.`ORDER_ID`)))  
  join `customers` `c` on((`o`.`CUSTOMER_ID` = `c`.`CUSTOMER_ID`)))  
  join `products` `p` on((`oi`.`PRODUCT_ID` = `p`.`PRODUCT_ID`)))  
group by `o`.`ORDER_ID`, `o`.`ORDER_DATETIME`, `o`.`ORDER_STATUS`, `c`.`CUSTOMER_ID`,  
  `c`.`EMAIL_ADDRESS`, `c`.`FULL_NAME`, `p`.`PRODUCT_NAME`,  
  `oi`.`LINE_ITEM_ID`, `oi`.`QUANTITY`, `oi`.`UNIT_PRICE` limit 10;
```

Task 3: Executando Queries com o HeatWave

- Não utilizando o HeatWave Cluster, o tempo de execução da query foi de aproximadamente 15.92 segundos:

```
Cloud Shell
MySQL 132.145.23.69:3306 ssl mysql_customer_orders SQL > select 'o'.'ORDER_ID' AS 'order_id', 'o'.'ORDER_DATETIME' AS 'ORDER_DATETIME',
-> 'o'.'ORDER_STATUS' AS 'order_status', 'c'.'CUSTOMER_ID' AS 'customer_id',
-> 'c'.'EMAIL_ADDRESS' AS 'email_address', 'c'.'FULL_NAME' AS 'full_name',
-> sum(('oi'.'QUANTITY' * 'oi'.'UNIT_PRICE')) AS 'order_total',
-> 'p'.'PRODUCT_NAME' AS 'product_name', 'oi'.'LINE_ITEM_ID' AS 'LINE_ITEM_ID',
-> 'oi'.'QUANTITY' AS 'QUANTITY', 'oi'.'UNIT_PRICE' AS 'UNIT_PRICE'
-> from (((('orders' `o` join `order_items` `oi` on (('o'.'ORDER_ID' = 'oi'.'ORDER_ID'))))
-> join `customers` `c` on (('o'.'CUSTOMER_ID' = 'c'.'CUSTOMER_ID'))))
-> join `products` `p` on (('oi'.'PRODUCT_ID' = 'p'.'PRODUCT_ID'))
-> group by 'o'.'ORDER_ID', 'o'.'ORDER_DATETIME', 'o'.'ORDER_STATUS', 'c'.'CUSTOMER_ID'
-> , 'c'.'EMAIL_ADDRESS', 'c'.'FULL_NAME', 'p'.'PRODUCT_NAME'
-> , 'oi'.'LINE_ITEM_ID', 'oi'.'QUANTITY', 'oi'.'UNIT_PRICE' limit 10;
```

order_id	ORDER_DATETIME	order_status	customer_id	email_address	full_name	order_total	product_name	LINE_ITEM_ID	QUANTITY	UNIT_PRICE
1	2022-02-04 00:00:00	COMPLETE	56	tamika.hu@internalmail	Tamika Hu	93.28	Boy's Pyjamas (Grey)	1	4	23.32
1	2022-02-04 00:00:00	COMPLETE	56	tamika.hu@internalmail	Tamika Hu	20.96	Boy's Shorts (Blue)	2	2	10.48
1	2022-02-04 00:00:00	COMPLETE	56	tamika.hu@internalmail	Tamika Hu	59.1	Boy's Shirt (White)	3	2	29.55
2	2022-02-08 00:00:00	COMPLETE	56	tamika.hu@internalmail	Tamika Hu	30.33	Women's Dress (Black)	1	3	10.11
2	2022-02-08 00:00:00	COMPLETE	56	tamika.hu@internalmail	Tamika Hu	185	Women's Jacket (Blue)	2	5	37.00
3	2022-02-09 00:00:00	COMPLETE	56	tamika.hu@internalmail	Tamika Hu	50.55	Women's Dress (Black)	1	5	10.11
4	2022-02-10 00:00:00	COMPLETE	56	tamika.hu@internalmail	Tamika Hu	76.68	Girl's Shorts (Green)	1	2	38.34
4	2022-02-10 00:00:00	COMPLETE	56	tamika.hu@internalmail	Tamika Hu	44	Girl's Pyjamas (Red)	2	4	11.00
4	2022-02-10 00:00:00	COMPLETE	56	tamika.hu@internalmail	Tamika Hu	156.64	Girl's Trousers (Red)	3	4	39.16
5	2022-02-11 00:00:00	COMPLETE	56	tamika.hu@internalmail	Tamika Hu	34.64	Girl's Pyjamas (Black)	1	4	8.66

```
10 rows in set (15.9225 sec)
MySQL 132.145.23.69:3306 ssl mysql_customer_orders SQL >
```

- Ou seja, com o motor HeatWave ativado, o tempo de execução da query foi 36x menor!**



Task 3: Executando Queries com o HeatWave

- Para manter o HeatWave habilitado, execute no Cloud Shell o comando abaixo:

```
SET SESSION use_secondary_engine=ON;
```

```
MySQL 132.145.23.69:3306 ssl mysql_customer_orders SQL > SET SESSION use_secondary_engine=ON;  
Query OK, 0 rows affected (0.0779 sec)  
MySQL 132.145.23.69:3306 ssl mysql_customer_orders SQL >
```

- Agora que você já viu como o HeatWave é capaz de acelerar suas consultas complexas e workloads analíticos, vamos ver na prática o funcionamento do LakeHouse!

Task 4: Executando Queries com CSV carregado no OCI Storage

- Para visualizar os arquivos CSV que estão no bucket:
- Em seu navegador, abra o link abaixo:
- objectstorage.uk-london-1.oraclecloud.com/p/J2eerVYTBE_2LTV8aRinz8S-HL0Rb7xjlyWOHDrTnQN_pzbpUUxxh41l1nj0YGSz/n/idazzjlcjqzj/b/LakeHouse_WS/o/

```
1 {  
2   "objects": [  
3     {  
4       "name": "order/delivery-orders-1.csv"  
5     },  
6     {  
7       "name": "order/delivery-orders-2.csv"  
8     },  
9     {  
10      "name": "order/delivery-orders-3.csv"  
11    }  
12  ]  
13 }
```

Task 4: Executando Queries com CSV carregado no OCI Storage

- Para a criação da tabela `delivery_orders`, utilizou-se o autoload para inferir o schema e estimar a capacidade necessária para a tabela “Delivery” no Object Store.
 - Parte das informações de ENTREGA dos pedidos está contida no arquivo `delivery-orders-1.csv`, que foi carregado no bucket.
- Criou-se uma tabela e carregou-se os dados do csv `delivery-orders-1.csv`:

```
MySQL 132.145.23.69:3306 ssl mysql_customer_orders SQL > CREATE TABLE `mysql_customer_orders`.`delivery_orders` ( `orders_delivery` int unsigned NOT NULL, `order_id` bigint unsigned NOT NULL, `customer_id` tinyint unsigned NOT NULL, `order_status` varchar(9) NOT NULL COMMENT 'RAPID_COLUMN=ENCODING=VARLEN', `store_id` tinyint unsigned NOT NULL, `delivery_vendor_id` tinyint unsigned NOT NULL, `estimated_time_hours` tinyint unsigned NOT NULL) ENGINE=lakehouse SECONDARY_ENGINE=RAPID ENGINE_ATTRIBUTE={'file': [{"par": "https://objectstorage.uk-1.oraclecloud.com/p/7T-9SoHuFWqeZaE4yqXOXYfOXxTQtzCkYPM8aUtXNHK-dCLz2gzEACa8K1kNd9Q-/n/idazzjlcjzj/b/LakeHouse_WS/o/order/delivery-orders-1.csv"}], "dialect": {"format": "csv", "delimiter": "\\t", "second_delimiter": "\\n"}};
```

- Visualização da tabela antes de inserirmos os dados do CSV no Object Store ->

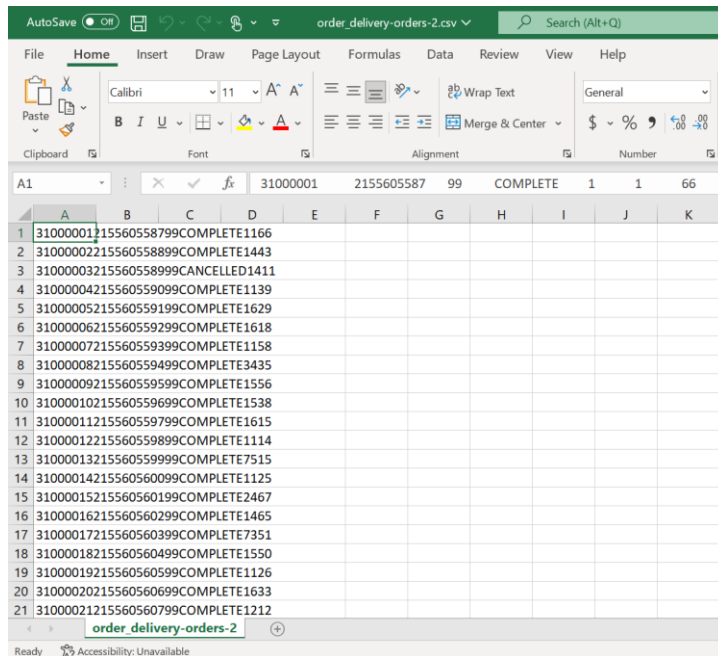
```
MySQL 10.0.1 33060+ ssl mysql_customer_orders SQL > desc delivery_orders;
```

Field	Type	Null	Key	Default	Extra
orders_delivery	int unsigned	NO		NULL	
order_id	bigint unsigned	NO		NULL	
customer_id	tinyint unsigned	NO		NULL	
order_status	varchar(9)	NO		NULL	
store_id	tinyint unsigned	NO		NULL	
delivery_vendor_id	tinyint unsigned	NO		NULL	
estimated_time_hours	tinyint unsigned	NO		NULL	

```
7 rows in set (0.0021 sec)
MySQL 10.0.1 33060+ ssl mysql_customer_orders SQL >
```

Task 4: Executando Queries com CSV carregado no OCI Storage

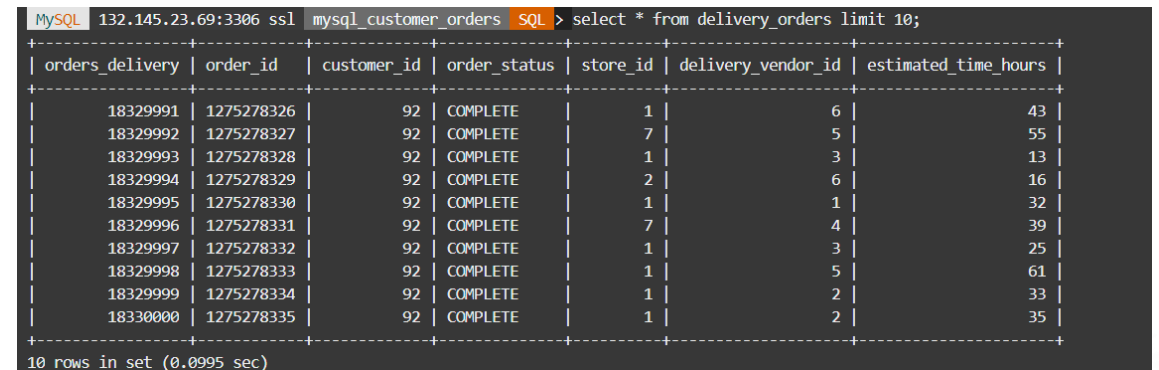
- Tabela delivery-orders originalmente em formato CSV:



	A	B	C	D	E	F	G	H	I	J	K
1	31000001	2155605587	99	COMPLETE	1	1	66				
2	31000002	2155605589	99	COMPLETE	1	1	66				
3	31000003	2155605590	99	CANCELLED	1	1	66				
4	31000004	2155605591	99	COMPLETE	1	1	66				
5	31000005	2155605592	99	COMPLETE	1	1	66				
6	31000006	2155605593	99	COMPLETE	1	1	66				
7	31000007	2155605594	99	COMPLETE	1	1	66				
8	31000008	2155605595	99	COMPLETE	1	1	66				
9	31000009	2155605596	99	COMPLETE	1	1	66				
10	31000010	2155605597	99	COMPLETE	1	1	66				
11	31000011	2155605598	99	COMPLETE	1	1	66				
12	31000012	2155605599	99	COMPLETE	1	1	66				
13	31000013	2155605600	99	COMPLETE	1	1	66				
14	31000014	2155605601	99	COMPLETE	1	1	66				
15	31000015	2155605602	99	COMPLETE	1	1	66				
16	31000016	2155605603	99	COMPLETE	1	1	66				
17	31000017	2155605604	99	COMPLETE	1	1	66				
18	31000018	2155605605	99	COMPLETE	1	1	66				
19	31000019	2155605606	99	COMPLETE	1	1	66				
20	31000020	2155605607	99	COMPLETE	1	1	66				
21	31000021	2155605608	99	COMPLETE	1	1	66				



- Tabela delivery-orders depois de espelhada em uma tabela:



```
MySQL 132.145.23.69:3306 ssl mysql_customer_orders SQL > select * from delivery_orders limit 10;
```

orders_delivery	order_id	customer_id	order_status	store_id	delivery_vendor_id	estimated_time_hours
18329991	1275278326	92	COMPLETE	1	6	43
18329992	1275278327	92	COMPLETE	7	5	55
18329993	1275278328	92	COMPLETE	1	3	13
18329994	1275278329	92	COMPLETE	2	6	16
18329995	1275278330	92	COMPLETE	1	1	32
18329996	1275278331	92	COMPLETE	7	4	39
18329997	1275278332	92	COMPLETE	1	3	25
18329998	1275278333	92	COMPLETE	1	5	61
18329999	1275278334	92	COMPLETE	1	2	33
18330000	1275278335	92	COMPLETE	1	2	35

10 rows in set (0.0995 sec)

Task 4: Executando Queries com CSV carregado no OCI Storage

- Para visualizar a quantidade de linhas da tabela, execute o comando abaixo:

```
select count(*) from delivery_orders;
```

- Para visualizar uma amostra dos dados da tabela, execute o comando abaixo:

```
select * from delivery_orders limit 5;
```

Task 4: Executando Queries com CSV carregado no OCI Storage

- Output dos comandos executados:

```
MySQL 132.145.23.69:3306 ssl mysql_customer_orders SQL > select count(*) from delivery_orders;
+-----+
| count(*) |
+-----+
| 29999998 |
+-----+
1 row in set (0.1011 sec)
MySQL 132.145.23.69:3306 ssl mysql_customer_orders SQL > select * from delivery_orders limit 5;
+-----+-----+-----+-----+-----+-----+-----+
| orders_delivery | order_id | customer_id | order_status | store_id | delivery_vendor_id | estimated_time_hours |
+-----+-----+-----+-----+-----+-----+-----+
| 10465777 | 726365484 | 88 | COMPLETE | 17 | 5 | 19 |
| 10465778 | 726365485 | 88 | COMPLETE | 18 | 2 | 53 |
| 10465779 | 726365486 | 88 | COMPLETE | 21 | 3 | 12 |
| 10465780 | 726365487 | 88 | COMPLETE | 21 | 3 | 17 |
| 10465781 | 726365488 | 88 | COMPLETE | 16 | 4 | 41 |
+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.1097 sec)
```


Task 4: Executando Queries com CSV carregado no OCI Storage

- Para executar um “join” da tabela delivery_orders, que foi criada a partir do CSV que estava no Object Storage com outra tabela que já estava carregada na base, execute o comando abaixo:

```
select o.*,d.* from orders o  
join delivery_orders d on o.order_id = d.order_id  
where o.order_id = 93751524;
```

- Output do comando acima:

```
MySQL 132.145.23.69:3306 ssl mysql_customer_orders SQL > select o.*,d.* from orders o  
-> join delivery_orders d on o.order_id = d.order_id  
-> where o.order_id = 93751524;
```

ORDER_ID	ORDER_DATETIME	CUSTOMER_ID	ORDER_STATUS	STORE_ID	orders_delivery	order_id	customer_id	order_status	store_id	delivery_vendor_id	estimated_time_hours
93751524	2022-08-23 00:00:00	81	COMPLETE	3	1377370	93751524	81	COMPLETE	3	3	65

- Pronto! Agora você conseguiu ver na prática como o HeatWave acelera as queries e também como é possível espelhar um CSV em uma tabela em questão de minutos utilizando o LakeHouse!**



Agradecimento 😊

- Agradecemos por participar do nosso WorkShop! Esperamos que tenha se divertido e também aprendido um pouco mais sobre HeatWave e LakeHouse.
- Para poder continuar a experiência com o MySQL HeatWave, escaneie o QR code abaixo e tenha acesso ao nosso portal com diversos Webinars gravados.

