UTM

UNIVERSITI TEKNOLOGI MALAYSIA

SCSP5023 - Big Data Management

Lab 1

Section 1

Ahmad Khaidir Amir bin Rodzman (A18CS0014)

Part A - Practice SQL DDL and DML statements

Table 1

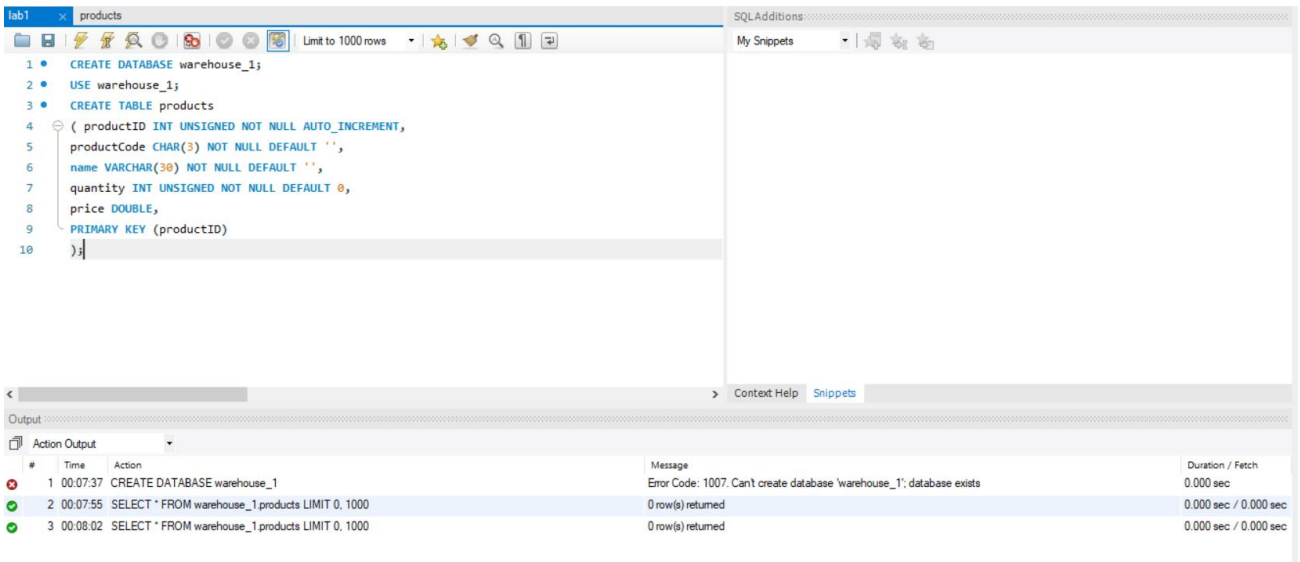
Product_ID	ProductCode	Name	Quantity	price
1001	Pen	Pen Red	5000	1.23
1002	Pen	Pen Blue	8000	1.25
1003	Pen	Pen Black	2000	1.25
1004	Pec	Pencil 2B	10000	0.49
1005	Pec	Pencil 2H	8000	0.48

1. Perform the following coding in MySQL Workbench. The following statement will create a database called warehouse1 and table called products

```
CREATE DATABASE warehouse1;
USE warehouse_1;
CREATE TABLE products
(
   productID INT UNSIGNED NOT NULL AUTO_INCREMENT,
   productCode CHAR(3) NOT NULL DEFAULT '',
   name VARCHAR(30) NOT NULL DEFAULT '',
   quantity INT UNSIGNED NOT NULL DEFAULT 0,
   price DOUBLE,
   PRIMARY KEY (productID)
   );
```

Explanation on data type of attributes

- 5 columns in the table products: productID, productCode, name, quantity and price. The types are:
- productID is INT UNSIGNED - non-negative integers.
- productCode is CHAR (3) - a fixed-length alphanumeric string of 3 characters.
- name is VARCHAR (30) - a variable-length string of up to 30 characters.
- quantity is also INT UNSIGNED (non-negative integers).
- price is DEOUBLE



```
1 •  SELECT * FROM warehouse_1.products;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

productID	productCode	name	quantity	price
NULL	NULL	NULL	NULL	NULL

2. The following insert statement will write data in Table 1 to products relation/table

```
INSERT INTO products VALUES (1001, 'PEN', 'Pen Red', 5000, 1.23);
INSERT INTO products VALUES
    (NULL, 'PEN', 'Pen Blue', 8000, 1.25),
    (NULL, 'PEN', 'Pen Black', 2000, 1.25);
INSERT INTO products (productCode, name, quantity, price) VALUES
    ('PEC', 'Pencil 2B', 10000, 0.48),
    ('PEC', 'Pencil 2H', 8000, 0.49);
```

NOTE: ProductID is an auto increment integer in which it will automatically increase +1 for the next record

Limit to 1000 rows | My Snippets

```
1 •  INSERT INTO products VALUES (1001, 'PEN', 'Pen Red', 5000, 1.23);
2    INSERT INTO products VALUES
3    (NULL, 'PEN', 'Pen Blue', 8000, 1.25),
4    (NULL, 'PEN', 'Pen Black', 2000, 1.25);
5 •  INSERT INTO products (productCode, name, quantity, price) VALUES
6    ('PEC', 'Pencil 2B', 10000, 0.48),
7    ('PEC', 'Pencil 2H', 8000, 0.49);
```

Context Help | Snippets

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	00:07:37	CREATE DATABASE warehouse_1	Error Code: 1007. Can't create database 'warehouse_1'; database exists	0.000 sec
2	00:07:55	SELECT * FROM warehouse_1.products LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
3	00:08:02	SELECT * FROM warehouse_1.products LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
4	00:16:03	INSERT INTO products VALUES (1001, 'PEN', 'Pen Red', 5000, 1.23)	1 row(s) affected	0.015 sec
5	00:16:03	INSERT INTO products VALUES (NULL, 'PEN', 'Pen Blue', 8000, 1.25), (NULL, 'PEN', 'Pen Black', 2000, 1.25)	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.000 sec
6	00:16:03	INSERT INTO products (productCode, name, quantity, price) VALUES ('PEC', 'Pencil 2B', 10000, 0.48), ('PEC', '...	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.000 sec
7	00:16:15	SELECT * FROM warehouse_1.products LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

```
1 •  SELECT * FROM warehouse_1.products;
```

Result Grid | Filter Rows: | Edit:

productID	productCode	name	quantity	price
1001	PEN	Pen Red	5000	1.23
1002	PEN	Pen Blue	8000	1.25
1003	PEN	Pen Black	2000	1.25
1004	PEC	Pencil 2B	10000	0.48
1005	PEC	Pencil 2H	8000	0.49
NULL	NULL	NULL	NULL	NULL

3. Perform the following quarries. Run one by one and observe the output

```
SELECT name, price FROM products;

SELECT * FROM products;

SELECT name, price FROM products WHERE price < 1.0;

SELECT name, price FROM products WHERE name LIKE 'PENCIL%';

SELECT * FROM products WHERE quantity >= 5000 AND name LIKE 'Pen %';

SELECT * FROM products
        WHERE (price BETWEEN 1.0 AND 2.0) AND (quantity BETWEEN 1000 AND 2000);

SELECT * FROM products ORDER BY price LIMIT 2;

SELECT * FROM products WHERE name LIKE 'Pen %' ORDER BY price DESC, quantity;

SELECT DISTINCT price, name FROM products;
```

```
1  SELECT name, price FROM products;
2  SELECT * FROM products;
3  SELECT name, price FROM products WHERE price < 1.0;
4  SELECT name, price FROM products WHERE name LIKE 'PENCIL%';
5  SELECT * FROM products WHERE quantity >= 5000 AND name LIKE 'Pen %';
6  SELECT * FROM products
7  WHERE (price BETWEEN 1.0 AND 2.0) AND (quantity BETWEEN 1000 AND 2000);
8  SELECT * FROM products ORDER BY price LIMIT 2;
9  SELECT * FROM products WHERE name LIKE 'Pen %' ORDER BY price DESC, quantity;
10 SELECT DISTINCT price, name FROM products;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	price
Pen Red	1.23
Pen Blue	1.25
Pen Black	1.25
Pencil 2B	0.48
Pencil 2H	0.49

Result Grid

Form Editor

productID	productCode	name	quantity	price
1001	PEN	Pen Red	5000	1.23
1002	PEN	Pen Blue	8000	1.25
1003	PEN	Pen Black	2000	1.25
1004	PEC	Pencil 2B	10000	0.48
1005	PEC	Pencil 2H	8000	0.49
NULL	NULL	NULL	NULL	NULL

name	price
Pencil 2B	0.48
Pencil 2H	0.49

productID	productCode	name	quantity	price
1004	PEC	Pencil 2B	10000	0.48
1005	PEC	Pencil 2H	8000	0.49
NULL	NULL	NULL	NULL	NULL

productID	productCode	name	quantity	price
1001	PEN	Pen Red	5000	1.23
1002	PEN	Pen Blue	8000	1.25
NULL	NULL	NULL	NULL	NULL

productID	productCode	name	quantity	price
1003	PEN	Pen Black	2000	1.25
NULL	NULL	NULL	NULL	NULL

	productID	productCode	name	quantity	price
▶	1004	PEC	Pencil 2B	10000	0.48
	1005	PEC	Pencil 2H	8000	0.49
*	NULL	NULL	NULL	NULL	NULL

	productID	productCode	name	quantity	price
▶	1003	PEN	Pen Black	2000	1.25
	1002	PEN	Pen Blue	8000	1.25
	1001	PEN	Pen Red	5000	1.23
*	NULL	NULL	NULL	NULL	NULL

	price	name
▶	1.23	Pen Red
	1.25	Pen Blue
	1.25	Pen Black
	0.48	Pencil 2B
	0.49	Pencil 2H

PART B – Import data file

1. Download **car93.cvs** file from the e-learning.
2. Observe the column heading and value in each column. The first column is the car type, 2nd column is price in thousand, 3rd column is fuel consumption in mile per gallon in the city, followed by drive train, no of seta and weight
3. Create an appropriate database, table, attribute name and type
4. Upload the data from **car93.cvs** into table created at step 3. Use *Table Data Import Wizard* (Right click the table created in step 3)
5. Perform the appropriate SQL DML (Select statement) to obtain the following: -
   a) List all rows of all columns
   b) List all the *midsize* type car with column type, price and weight.
   c) List category of *drive_train*
   d) List all large car with 6 seaters in ascending order according to price.
   e) List all car that have weight >2000
   f) List the first ten record in the table in descending order according to mpg_city
   g) List all car that having price between 25 to 35 (thousand)
   h) List all large car with mpg_city < 20
   i) List the midsize car type with *drive_train* front

Submission instruction.
You only need to submit the lab exercise in Part B. Include query and output of query statement in the submission.

```
1  CREATE DATABASE car93;
2  USE car93;
3  CREATE TABLE car
4  (
5  carID INT UNSIGNED NOT NULL AUTO_INCREMENT,
6  type CHAR(3) NOT NULL DEFAULT '',
7  price DOUBLE,
8  mpg_city INT UNSIGNED NOT NULL DEFAULT 0,
9  drive_train VARCHAR(30) NOT NULL DEFAULT '',
10 passengers INT UNSIGNED NOT NULL DEFAULT 0,
11 weight INT UNSIGNED NOT NULL DEFAULT 0,
12 PRIMARY KEY (carID)
13 );
```

	carID	type	price	mpg_city	drive_train	passengers	weight
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

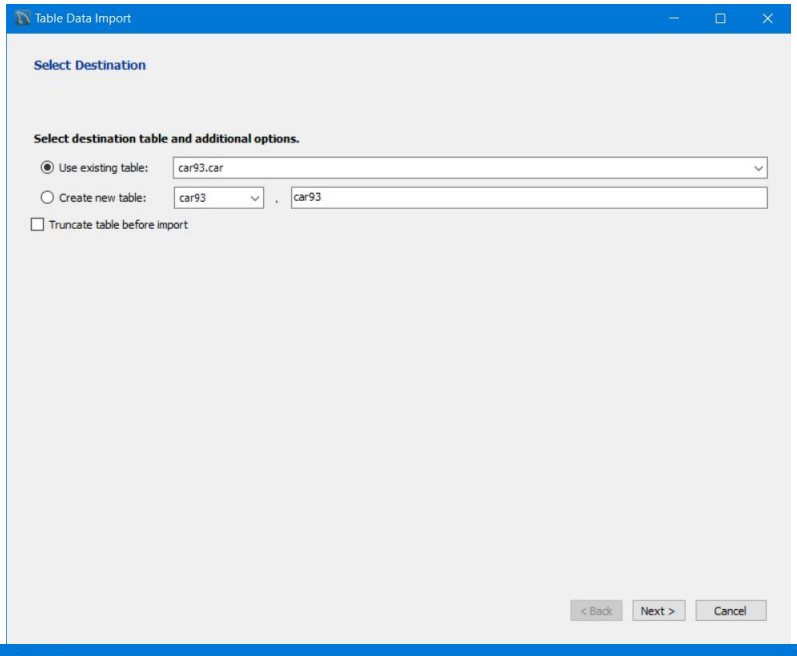


Table Data Import
Select Destination
Select destination table and additional options.
Use existing table:
car93.car
Create new table:
car93
car93
Truncate table before import
< Back
Next >
Cancel

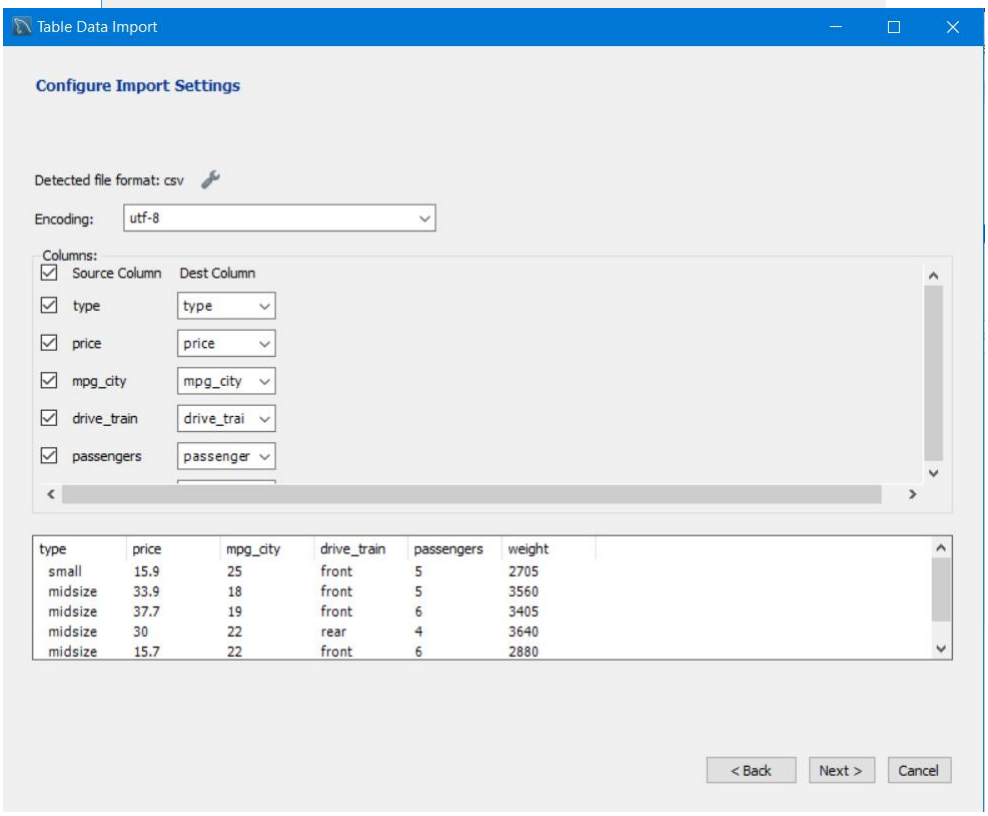
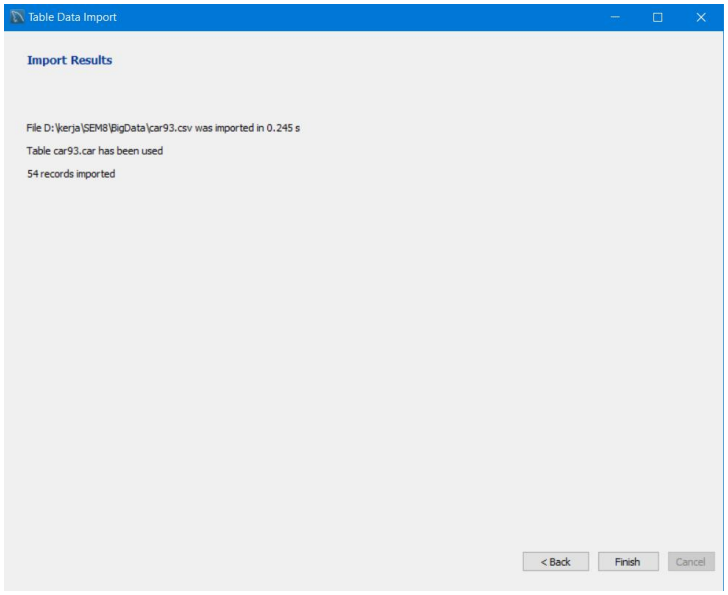
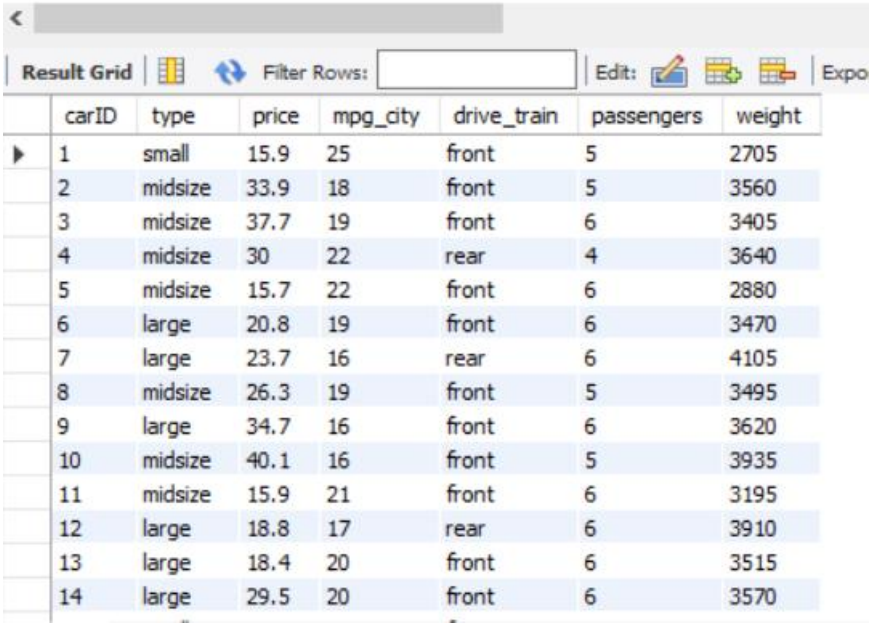


Table Data Import
Configure Import Settings
Detected file format: csv
Encoding:
utf-8
Columns:
Source Column
Dest Column
type
type
price
price
mpg_city
mpg_city
drive_train
drive_trai
passengers
passenger
type price mpg_city drive_train passengers weight
small 15.9 25 front 5 2705
midsize 33.9 18 front 5 3560
midsize 37.7 19 front 6 3405
midsize 30 22 rear 4 3640
midsize 15.7 22 front 6 2880
< Back
Next >
Cancel



a) List all rows of all columns



Result Grid | Filter Rows: | Edit: | Expo

carID	type	price	mpg_city	drive_train	passengers	weight
1	small	15.9	25	front	5	2705
2	midsize	33.9	18	front	5	3560
3	midsize	37.7	19	front	6	3405
4	midsize	30	22	rear	4	3640
5	midsize	15.7	22	front	6	2880
6	large	20.8	19	front	6	3470
7	large	23.7	16	rear	6	4105
8	midsize	26.3	19	front	5	3495
9	large	34.7	16	front	6	3620
10	midsize	40.1	16	front	5	3935
11	midsize	15.9	21	front	6	3195
12	large	18.8	17	rear	6	3910
13	large	18.4	20	front	6	3515
14	large	29.5	20	front	6	3570

b) List all the *midsize* type car with column type, price and weight.

```
1    SELECT type, price, weight FROM car WHERE type LIKE 'midsize%';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

type	price	weight
midsize	33.9	3560
midsize	37.7	3405
midsize	30	3640
midsize	15.7	2880
midsize	26.3	3495
midsize	40.1	3935
midsize	15.9	3195
midsize	15.6	3080
midsize	20.2	3325
midsize	13.9	2885
midsize	47.9	4000
midsize	28	3510
midsize	35.2	3515
midsize	34.3	3695
midsize	61.9	3525
midsize	14.9	3610
midsize	26.1	3730
midsize	21.5	3200
midsize	16.3	2890
midsize	18.5	3450
midsize	18.2	3030
midsize	26.7	3245

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

c) List category of *drive_train*

```
1 •    SELECT  DISTINCT drive_train FROM car;
```

Result Grid | Filter Rows: | Export:

drive_train
front
rear
4WD

d) List all large car with 6 seaters in ascending order according to price.

```
1    SELECT * FROM car WHERE passengers = 6 AND type LIKE "large" ORDER BY price ASC
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

carID	type	price	mpg_city	drive_train	passengers	weight
13	large	18.4	20	front	6	3515
12	large	18.8	17	rear	6	3910
19	large	19.3	20	front	6	3490
43	large	20.7	19	front	6	3470
6	large	20.8	19	front	6	3470
23	large	20.9	18	rear	6	3950
7	large	23.7	16	rear	6	4105
46	large	24.4	19	front	6	3495
14	large	29.5	20	front	6	3570
9	large	34.7	16	front	6	3620
33	large	36.1	18	rear	6	4055
NULL	NULL	NULL	NULL	NULL	NULL	NULL

e) List all car that have weight >2000

```
1    SELECT * FROM car WHERE weight >= 2000
```

Result Grid | Filter Rows: | Edit: | Exp

carID	type	price	mpg_city	drive_train	passengers	weight
1	small	15.9	25	front	5	2705
2	midsize	33.9	18	front	5	3560
3	midsize	37.7	19	front	6	3405
4	midsize	30	22	rear	4	3640
5	midsize	15.7	22	front	6	2880
6	large	20.8	19	front	6	3470
7	large	23.7	16	rear	6	4105
8	midsize	26.3	19	front	5	3495
9	large	34.7	16	front	6	3620
10	midsize	40.1	16	front	5	3935
11	midsize	15.9	21	front	6	3195
12	large	18.8	17	rear	6	3910
13	large	18.4	20	front	6	3515
14	large	29.5	20	front	6	3570
15	small	9.2	29	front	5	2270
16	small	11.3	23	front	5	2670
17	midsize	15.6	21	front	6	3080
18	small	12.2	29	front	5	2295
19	large	19.3	20	front	6	3490
21	small	10.1	23	front	5	2530
22	midsize	20.2	21	front	5	3325
23	large	20.9	18	rear	6	3950

f) List the first ten record in the table in descending order according to mpg_city

```
1    SELECT * FROM car  ORDER BY mpg_city DESC LIMIT 10;
```

Result Grid | Filter Rows: | Edit: | Export/I

carID	type	price	mpg_city	drive_train	passengers	weight
24	small	8.4	46	front	4	1695
25	small	12.1	42	front	4	2350
50	small	8.6	39	front	4	1965
48	small	8.4	33	4WD	4	2045
51	small	9.8	32	front	5	2055
44	small	9	31	front	4	2350
20	small	7.4	31	front	4	1845
18	small	12.2	29	front	5	2295
26	small	8	29	front	5	2345
15	small	9.2	29	front	5	2270
NULL	NULL	NULL	NULL	NULL	NULL	NULL

g) List all car that having price between 25 to 35 (thousand)

```
1    SELECT * FROM car
2    WHERE (price BETWEEN 25.0 AND 35.0);
```

Result Grid | Filter Rows: | Edit:

carID	type	price	mpg_city	drive_train	passengers	weight
2	midsize	33.9	18	front	5	3560
4	midsize	30	22	rear	4	3640
8	midsize	26.3	19	front	5	3495
9	large	34.7	16	front	6	3620
14	large	29.5	20	front	6	3570
30	midsize	28	18	front	5	3510
32	midsize	34.3	17	front	6	3695
39	midsize	26.1	18	front	5	3730
54	midsize	26.7	20	front	5	3245
NULL	NULL	NULL	NULL	NULL	NULL	NULL

h) List all large car with mpg_city < 20

```
1    SELECT * FROM car WHERE mpg_city <= 20 AND type LIKE 'large';
```

Result Grid | Filter Rows: | Edit: | Export/Import:

carID	type	price	mpg_city	drive_train	passengers	weight
6	large	20.8	19	front	6	3470
7	large	23.7	16	rear	6	4105
9	large	34.7	16	front	6	3620
12	large	18.8	17	rear	6	3910
13	large	18.4	20	front	6	3515
14	large	29.5	20	front	6	3570
19	large	19.3	20	front	6	3490
23	large	20.9	18	rear	6	3950
33	large	36.1	18	rear	6	4055
43	large	20.7	19	front	6	3470
46	large	24.4	19	front	6	3495
NULL	NULL	NULL	NULL	NULL	NULL	NULL

i) List the midsize car type with *drive_train* front

```
1    SELECT * FROM car WHERE type LIKE 'midsize' AND drive_train LIKE 'front';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell C

carID	type	price	mpg_city	drive_train	passengers	weight
2	midsize	33.9	18	front	5	3560
3	midsize	37.7	19	front	6	3405
5	midsize	15.7	22	front	6	2880
8	midsize	26.3	19	front	5	3495
10	midsize	40.1	16	front	5	3935
11	midsize	15.9	21	front	6	3195
17	midsize	15.6	21	front	6	3080
22	midsize	20.2	21	front	5	3325
28	midsize	13.9	20	front	5	2885
30	midsize	28	18	front	5	3510
32	midsize	34.3	17	front	6	3695
39	midsize	26.1	18	front	5	3730
41	midsize	21.5	21	front	5	3200
42	midsize	16.3	23	front	5	2890
45	midsize	18.5	19	front	5	3450
52	midsize	18.2	22	front	5	3030
54	midsize	26.7	20	front	5	3245
NULL	NULL	NULL	NULL	NULL	NULL	NULL