SQL - Revision

Assoc. Prof. David Taniar

SQL - Revision

Outline:

A. Introduction to SQL client environment
B. Create tables
C. Insert into
D. Simple query retrieval
E. Updating and deleting records
F. Commit
G. Joining multiple tables
H. Aggregate functions and group by
I. Alter tables

A. Introduction to SQL client

- **There are several SQL client software:**

  - SQL Developer

  - SQL*Plus (Windows, Mac, Unix)

- **Login details:**

  - Username: S12345678 (12345678 is your student id)

  - Password: student

  - Host string: the unit code

Introduction to SQL Developer

Introduction to SQL Developer

Single Statement Execution

Script Execution

Type Command

Output of Execution

B. Create Tables

- General Syntax:

    CREATE TABLE <table_name>

    (attribute1   **data_type**   <NOT NULL>,

    attribute2   **data_type**,

    …

    PRIMARY KEY (attribute1),

    FOREIGN KEY (attribute1)

    REFERENCES <references table_name> (key_attribute));

Create Tables – Data Types

- Data type denotes a kind of data of an attribute value

  - Character data types: VARCHAR2 and CHAR

    Make VARCHAR2(20)
    Model VARCHAR2(30)

  - Number data types: NUMBER

    Year NUMBER(4)
    StampDuty NUMBER(8,2)

  - Date data type: DATE

    Salesdate DATE

Create Tables – Data Types

- <u>Date Data Type</u>

  - DATE stores dates from 1/1/4712 BC to 12/31/4712 AD

  - Default date format: DD-MON-YYYY

    - example: 05-JUN-2011

  - Example declaration: *salesDate DATE*

  - DATE data type also stores time values

Create Tables – Data Types

- Default **time** format: HH:MI:SS A.M.

    - If no time value is given when a date is inserted, default value is

      12:00:00 A.M.

    - If no date value is given when a time is inserted, default date is

      first day of current month

    - Example salesDate field: 07-JUN-2016 12:00:00 A.M.

Create Tables – Constraints

- Integrity Constraints

  - Primary Key attribute

  - NOT NULL constraints

    - Specifies that a field cannot be NULL

    - Sample Declaration: *Field_name   data_type   NOT NULL*

  - Foreign Key attribute in a table refers to another record in another table

Create Tables – Example 1

- **Example:** *create CAR, CARSALES and CUSTOMER tables*
  - Commands:
    - CAR table

```
CREATE TABLE Car
(carID          NUMBER(5) NOT NULL,
 make           VARCHAR2(20) NOT NULL,
 model          VARCHAR2(30),
 year           NUMBER(4),
 colour         VARCHAR2(25),
 advertisedPrice NUMBER(10),
 PRIMARY KEY (carID)
);
```

CARSALES

CarID
CustomerID
SalesDate
PurchasedPrice
StampDuty

CAR

CarID
Make
Model
Year
Colour
advertisedPrice

CUSTOMER

CustomerID
CName
CPhone
Address
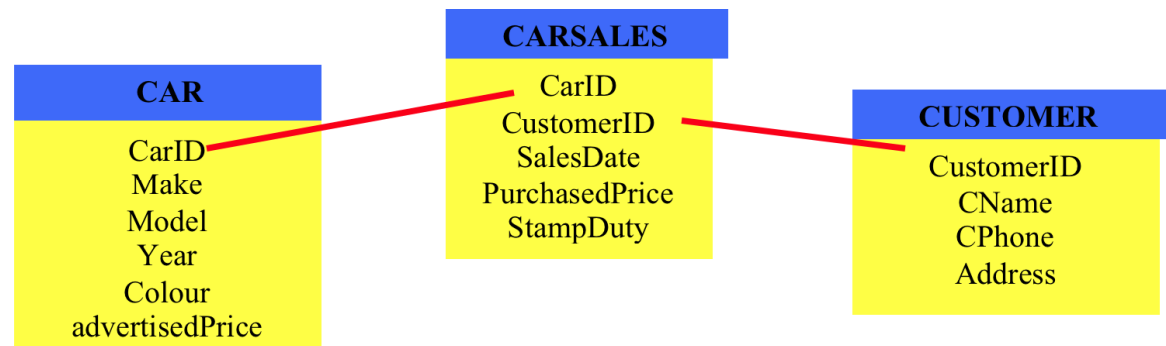
Create Tables – Example 2

- <u>CUSTOMER table</u>

```
CREATE TABLE Customer

(customerID NUMBER(5) NOT NULL,

 cNAME       VARCHAR2(20) NOT NULL,

 cPhone      VARCHAR2(10),

 address     VARCHAR2(50),

 PRIMARY KEY (customerID)

);
```

CAR

CarID
Make
Model
Year
Colour
advertisedPrice

CARSALES

CarID
CustomerID
SalesDate
PurchasedPrice
StampDuty

CUSTOMER

CustomerID
CName
CPhone
Address

Create Tables – Example 3

- <u>CARSALES table</u>

```
CREATE TABLE Carsales
(carID           NUMBER(5) NOT NULL,
 customerID      NUMBER(5) NOT NULL,
 salesDate       DATE NOT NULL,
 purchasedPrice  NUMBER(10) NOT NULL,
 stampDuty       NUMBER(8,2) NOT NULL,
 PRIMARY KEY (carID,customerID),
 FOREIGN KEY (carID) REFERENCES Car(carID),
 FOREIGN KEY (customerID) REFERENCES Customer(customerID)
);
```

CAR

CarID
Make
Model
Year
Colour
advertisedPrice

CARSALES

CarID
CustomerID
SalesDate
PurchasedPrice
StampDuty

CUSTOMER

CustomerID
CName
CPhone
Address
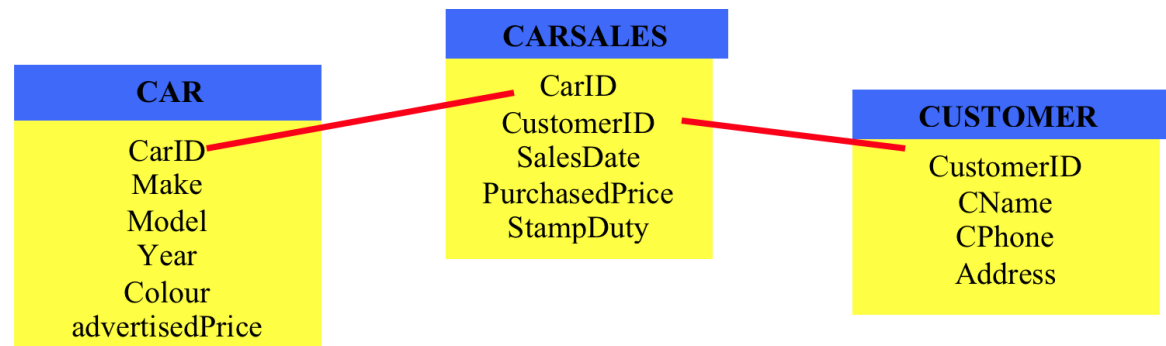
Create Tables – by Copying

- You can create a new table by copying from an existing table:

    CREATE TABLE <table_name>

    AS SELECT *

    FROM …   ;

- For example:

    CREATE TABLE car

    AS SELECT *

    FROM dtaniar.car;

- Notes:

    - It creates and copy the records from the existing table

    - However, it does not copy the PK and FK

    - In the above example, it copies the table from dtaniar account. **15**

Create Tables – View Tables

- ## Viewing Information about Tables

  - To view <u>all tables</u> in the database, the general syntax is:

    SELECT * FROM TAB;

    - For example:

      SELECT * FROM tab;

      ```
      TNAME                             TABTYPE   CLUSTERID

      ------------------------------    -------   ----------

      CAR                               TABLE

      CARSALES                          TABLE

      CUSTOMER                          TABLE
      ```

Create Tables – Describe Tables

- To view the <u>table structure</u>, the general syntax is:

  DESCRIBE <table_name>

  - For example:

  DESC car;

  ```
  Name                  Null       Type

  --------------- -------- -----------

  CARID                 NOT NULL NUMBER(5)

  MAKE                  NOT NULL VARCHAR2(20)

  MODEL                          VARCHAR2(30)

  YEAR                           NUMBER(4)

  COLOUR                         VARCHAR2(25)

  ADVERTISEDPRICE                NUMBER(10)
  ```
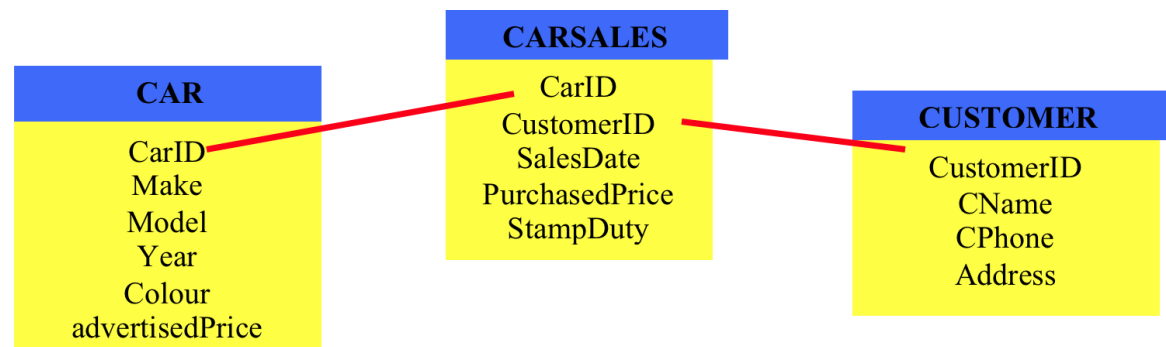
Create Tables – Drop Tables

- To <u>drop</u> an unwanted table, the general syntax is:

  DROP TABLE <table_name>;

  - For example:

  DROP TABLE Car;

    - If the table that you want to delete (e.g. table Car) is being used as a FK by another table (e.g. table Carsales), then you cannot delete table Car.
    - In this case, you need to delete table Carsales first, before deleting table Car.

C. Insert Into

- **General Syntax**

  - To insert values for every single attribute in a record:

    INSERT INTO <table_name>

    VALUES (attribute1_value, attribute2_value,….);

    - **Example:**

      INSERT INTO Car

      VALUES (1, 'Holden', 'Cruze', 2015, 'Black', 25780);

  - Strings are enclosed in single quotes (') and are case-sensitive
    (e.g. 'General Practice' is different from 'general practice')

Insert Into

- To insert a value of selected attributes:

  INSERT INTO <table_name> (attribute1,attribute2,…)

  VALUES (attribute1_value, attribute2_value,….)

  - **Example:**

    INSERT INTO Car (carID, make) VALUES ('16', 'Audi');

Insert Into

- The TO_DATE function:

  - **TO_DATE** ('date_value', 'format mask');

  - **Example:**

    INSERT INTO Carsales

    *VALUES (1,4, **TO_DATE('04/Feb/2015', 'DD/MON/YYYY')**,25780,824.96);*

Insert Into

- The common DATE format masks

Format Mask	Formatted Data
DD-MON-YYYY	05-FEB-2007
MM/DD/YYYY	02/05/2007
HH:MI AM	02:30 PM
MONTH DAY, YYYY	FEB 5, 2007
MM/DD/YYYY HH:MI AM	02/05/2007 02:30 PM

Sample DATE format masks

Insert Into

- Insert multiple records **one-by-one**:

  INSERT INTO Car VALUES (2, 'BMW', '520d', 2016, 'Grey', 98800);
  INSERT INTO Car VALUES (3, 'Audi', 'A5', 2016, 'Black', 68200);
  INSERT INTO Car VALUES (4, 'Holden', 'Commodore', 2008, 'Grey', 12650);

- Insert multiple records at **once**:

  INSERT ALL
  INTO Car VALUES (2, 'BMW', '520d', 2016, 'Grey', 98800)
  INTO Car VALUES (3, 'Audi', 'A5', 2016, 'Black', 68200)
  INTO Car VALUES (' 4, 'Holden', 'Commodore', 2008, 'Grey', 12650)
  *SELECT * FROM DUAL;*

D. Simple Query Retrieval

- ## Simple Retrieval

  - **Retrieve all Records**

    - **General Syntax :**

      SELECT *

      FROM <table_name>;

- Output:

```
CARID MAKE           MODEL           YEAR   COLOUR   ADVERTISEDPRICE
----- -------------- --------------- -----  -------  ----------------
    1 Holden         Cruze           2015   Black              25780
    2 BMW            520d            2016   Grey               98800
    3 Audi           A5              2016   Black              68200
    4 Holden         Commodore       2008   Grey               12650
    5 Honda          Civic           2012   Red                13488
    6 Jeep           Grand Cherokee  2006   Black              21999
    7 Lexus          RX350           2009   Red                33500
    8 Mazda          3Series         2011   Blue               13999
    9 Nissan         Dualis          2010   White              18888
   10 Volkwagen      Golf            2015   White              39888
   11 Mercedes Benz  C200            2006   Blue               16995
   12 Subaru         Outback         2014   null               33950
   13 Honda          City            2010   Grey                7490
   14 Mini           Cooper          2009   Black              19750
   15 Toyota         Corolla         2013   White              23888
15 rows selected
```

  - **Example:** retrieve everything from the CAR table:

    *SELECT * FROM car;*

Simple Query Retrieval

- Output:
  ```
  MODEL
  ------------------
  Cruze
  520d
  A5
  Commodore
  Civic
  Grand Cherokee
  RX350
  3Series
  Dualis
  Golf
  C200
  Outback
  City
  Cooper
  Corolla

   15 rows selected
  ```

- **Retrieve Specific Fields**

  - **General Syntax:**

    SELECT <attribute1, attribute2…>

    FROM <table_name>;

  - **Example:** *select only the model from the CAR table*

    SELECT model FROM car;

Simple Query Retrieval

- **Eliminating Duplicated Records** *(DISTINCT qualifier)*

  - **General Syntax:**

    SELECT DISTINCT <attribute1,attribute2,…>

    FROM <table_name>;

  - **Example:** eliminating duplicates for the MAKE values:

    SELECT DISTINCT make FROM car;

Simple Query Retrieval

- Original Output (without DISTINCT):

```
MAKE
_____
Holden
BMW
Audi
Holden
Honda
Jeep
Lexus
Mazda
Nissan
Volkwagen
Mercedes Benz
Subaru
Honda
Mini
Toyota

15 rows selected
```

- Output (with DISTINCT):

```
MAKE
_____
Holden
Lexus
Subaru
BMW
Mazda
Nissan
Audi
Volkwagen
Toyota
Honda
Mercedes Benz
Jeep
Mini

13 rows selected
```

Simple Query Retrieval

- **DISTINCT Multiple Attributes**

  - **Example:** eliminating duplicates for the MAKE and YEAR values:

*SELECT **DISTINCT** make, year*

FROM car;

```
MAKE                           YEAR
-------------------- ----------
Volkwagen                      2015
Toyota                         2013
Mini                           2009
Holden                         2015
BMW                            2016
Honda                          2012
Subaru                         2014
Holden                         2008
Jeep                           2006
Mazda                          2011
Audi                           2016
Lexus                          2009
Honda                          2010
Nissan                         2010
Mercedes Benz                  2006

 15 rows selected
```

Simple Query Retrieval

- **Conditional Retrieval**

  - Search Conditions specified for more complex data retrieval

  - **The WHERE Clause**

    - Operators:

      - equal (=)

      - greater than (>)

      - less than (<)

      - greater than or equal to (>=)

      - less than or equal to (<=)

      - Not equal (<>)

Simple Query Retrieval

- **General Syntax:**

  SELECT <attribute1,attribute2,...>

  FROM <ownername.table_name1>

  WHERE <search condition>;

- **Example:**

  SELECT year, model, advertisedPrice

  FROM car

  WHERE make='Holden'

YEAR	MODEL	ADVERTISEDPRICE
2015	Cruze	25780
2008	Commodore	12650

Simple Query Retrieval

- **"BETWEEN"**

    - **Example:** *list the carid, salesDate and purchasedPrice after 2014 and before 2016*

        SELECT carid, salesdate, purchasedPrice

        FROM carsales

        WHERE salesdate

        BETWEEN TO_DATE('01-JAN-2014', 'DD-MON-YYYY')

        AND TO_DATE('31-DEC-2015','DD-MON-YYYY');

```
   CARID SALESDATE    PURCHASEDPRICE
---------- ----------- --------------
        1 04-FEB-2015          25780
        8 12-DEC-2015          13999
```

Simple Query Retrieval

- **Using a String Comparison**

    - **Example:** *list the carid, salesDate and purchasedPrice after 2014 and before 2016*

        SELECT carid, salesdate, purchasedPrice

        FROM carsales

        WHERE TO_CHAR(salesdate, 'YYYYMMDD') >'20140101'

        AND TO_CHAR(salesdate, 'YYYY'MMDD') < '20151231';

```
 CARID SALESDATE    PURCHASEDPRICE
---------- ----------- --------------
     1 04-FEB-2015          25780
     8 12-DEC-2015          13999
```

Simple Query Retrieval

- **"AND" or "OR"**

  - AND: both conditions must be true

    - **Example:**

    SELECT make, model, year

    FROM car

    *WHERE year>2014 **AND** make='Holden';*

```
MAKE        MODEL              YEAR

--------- ---------------- ----------

Holden    Cruze               2015
```

Simple Query Retrieval

- OR: either one of the condition is true
    - **Example:**

        SELECT make, model, colour, year, advertisedPrice

        FROM car

        WHERE colour='Red' OR year>2014;

```
MAKE       MODEL      COLOUR           YEAR ADVERTISEDPRICE
---------  ---------- ---------- ---------- ----------------
Holden     Cruze      Black            2015            25780
BMW        520d       Grey             2016            98800
Audi       A5         Black            2016            68200
Honda      Civic      Red              2012            13488
Lexus      RX350      Red              2009            33500
Volkwagen  Golf       White            2015            39888
  6 rows selected
```

Simple Query Retrieval

- ## Other Conditions

  - ### LIKE/NOT LIKE

    - **Example:** *displaying all MAKE from car table that has their first character as 'M'*

      SELECT make

      FROM CAR

      *WHERE make **LIKE** 'M%';*

      ```
      MAKE
      ----------------
      Mazda
      Mercedes Benz
      Mini
      ```

Simple Query Retrieval

- **IN/NOT**
  - suitable to perform a set member search
  - **Example:**
    - IN - *displaying all MODEL that its MAKE is either 'Holden' or 'Honda'*
      SELECT make, model
      FROM car
      *WHERE make **IN** ('Holden', 'Honda');*

```
MAKE                      MODEL
--------------------      ------------
Holden                    Cruze
Holden                    Commodore
Honda                     Civic
Honda                     City
```

Simple Query Retrieval

- **IN/NOT**
  - **Example:**
    - NOT IN *- displaying make, model and colour of cars that are not 'Black' and 'White'*

      SELECT make, model, colour

      FROM car

      WHERE colour **NOT IN** *('Black', 'White');*

      ```
      MAKE            MODEL        COLOUR
      --------------  -----------  ------------
      BMW             520d         Grey
      Holden          Commodore    Grey
      Honda           Civic        Red
      Lexus           RX350        Red
      Mazda           3Series      Blue
      Mercedes Benz   C200         Blue
      Honda           City         Grey
      ```

Simple Query Retrieval

- **NULL/NOT NULL**

  - **Example:**

    - NULL Operator:

      SELECT carid, make

      FROM car

      *WHERE colour **IS NULL**;*

```
CARID  MAKE

------ ---------

    12 Subaru
```

Simple Query Retrieval

- NOT NULL Operator:

SELECT carid, make
FROM car
*WHERE colour **IS NOT NULL**;*

```
 CARID MAKE
------- --------------------
      1 Holden
      2 BMW
      3 Audi
      4 Holden
      5 Honda
      6 Jeep
      7 Lexus
      8 Mazda
      9 Nissan
     10 Volkwagen
     11 Mercedes Benz
     13 Honda
     14 Mini
     15 Toyota
14 rows selected
```

Simple Query Retrieval

- **MULTIPLE OPERATORs**

  - **Example:** *list cars make start with 'M' and was made before 2016*

    SELECT year, make, model

    FROM car

    WHERE make LIKE 'M%'

    AND year<2016;

```
YEAR MAKE             MODEL

---- ---------------- -------------

2011 Mazda            3Series

2006 Mercedes Benz    C200

2009 Mini             Cooper
```

Simple Query Retrieval

- ## Sorting

  - specify to sort the output by using ORDER BY

  - **General Syntax:**

    SELECT <attribute1,attribtue2,..>

    FROM <table_name>

    ORDER BY <attribute_name> [**DESC**];

Simple Query Retrieval

- **Example:** *retrieving all make, model, advertisedPrice and colour in a descending order of the advertisedPrice*

  SELECT make, model, advertisedPrice, colour
  FROM car
  ORDER BY *advertisedPrice* **DESC**;

```
MAKE              MODEL              ADVERTISEDPRICE   COLOUR
----------------  -----------------  ---------------   --------
BMW               520d               98800             Grey
Audi              A5                 68200             Black
Volkwagen         Golf               39888             White
Subaru            Outback            33950             NULL
Lexus             RX350              33500             Red
Holden            Cruze              25780             Black
Toyota            Corolla            23888             White
Jeep              Grand Cherokee     21999             Black
Mini              Cooper             19750             Black
Nissan            Dualis             18888             White
Mercedes Benz     C200               16995             Blue
Mazda             3Series            13999             Blue
Honda             Civic              13488             Grey
Holden            Commodore          12650             Grey
Honda             City                7490             Grey
15 rows selected
```

Simple Query Retrieval

- **"AS":** to rename a column

  - **Example:** *list all SalesDate, PurchasedPrice, StampDuty and TotalPrice (TotalPrice = PurchasedPrice+StampDuty)*

    SELECT salesdate, purchasedPrice, stampduty,

    (purchasedPrice+StampDuty) as TotalPrice

    FROM carSales;

```
SALESDATE     PURCHASEDPRICE   STAMPDUTY   TOTALPRICE
-----------   --------------   ---------   ----------
04-FEB-2015            25780      824.96     26604.96
13-JUL-2016            12650         506        13156
12-DEC-2015            13999      559.96     14558.96
14-JUN-2016            39888     1276.42     41164.42
18-MAY-2016            98800      5631.6     104431.6
```

E. Updating and Deleting Records

- In the created tables

    - **UPDATE** command – updating

    - **DELETE** command – deletion

Updating Records

- ## Update

  - ### General Syntax:

    UPDATE <table_name>

    SET <attribute_name> = <new_value>

    WHERE <expression> <operator> <expression>;

    - records can be updated in only one table at a time

    - update multiple fields that are within the same table

    - WHERE clause - make the command updates specific records only

Updating Records

- **Example:**

  - *Updating colour of carID '5' from 'Red' to 'Grey'*

    UPDATE car

    SET colour='Grey'

    WHERE carID =5;

Deleting Records

- ## Delete

  - ### General Syntax:

    DELETE FROM <table_name>

    WHERE <search_condition>;

    - remove specific records from a database table

    - use WHERE clause to specify multiple records to delete multiple records at one time

    - If the search condition is omitted, all records in the table are deleted.

Deleting Records

- **Example:**

    - deleting a single record from the CAR table

        DELETE FROM car

        WHERE carid= 5;

    - deleting multiple records from the CAR table that contain MAKE starting with 'M'

        DELETE FROM car

        WHERE make LIKE 'M%';

    - deleting all records from the CAR table

        DELETE FROM car;

Deleting Records

- **Notice:**
  - not allowed to delete a primary key record that has its corresponded foreign key somewhere else in another table
  - **Example:** *delete a primary key CarId is 5 in the CAR table that has a foreign key record that CarId is 5 in the CARSALES table*

    DELETE FROM car WHERE carid=5;

    DELETE FROM car

    *

    ERROR at line 1:

    ORA-02292: integrity constraint (SYSTEM.SYS_C005453) violated - child record found

F. Commit

- **Commit**

  - When inserted data by issuing the INSERT command

    - the changes are only saved in the local database buffer

    - are not saved in the database

    - until you COMMIT the transaction

  - *it is important to remember to COMMIT whenever you have finished inserting values or make changes to the database values*

Commit

- **General Syntax:**
  - Sample of inserting a new record with commit

```
CARID MAKE            MODEL          YEAR COLOUR   ADVERTISEDPRICE
----- --------------- ------------ ---------- ---------  ----------------
    1 Holden          Cruze          2015 Black              25780
    2 BMW             520d           2016 Grey               98800
    3 Audi            A5             2016 Black              68200
    4 Holden          Commodore      2008 Grey               12650
    5 Honda           Civic          2012 Red                13488
    6 Jeep            Grand Cherokee 2006 Black              21999
    7 Lexus           RX350          2009 Red                33500
    8 Mazda           3Series        2011 Blue               13999
    9 Nissan          Dualis         2010 White              18888
   10 Volkwagen       Golf           2015 White              39888
   11 Mercedes Benz   C200           2006 Blue               16995
   12 Subaru          Outback        2014 null               33950
   13 Honda           City           2010 Grey                7490
   14 Mini            Cooper         2009 Black              19750
   15 Toyota          Corolla        2013 White              23888
 15 rows selected
SQL> INSERT INTO CAR
  2  VALUES(16,'Holden', 'Cruze', 2016, 'Black', 27800);
1 row created;

SQL> commit;

Commit complete;
```

All original records are currently in CAR table

Insert a new record '16' with commit

Effect of Inserting New Record with Commit

G. Joining Multiple Tables

- ## Join

  - database query to join multiple database tables together

    - the data needed or the conditions specified come from more than one table.

  - **Syntax:**

    SELECT <attribute1,attribute2,… >

    FROM <table_name1,table_name2,…>

    WHERE <table_name1.join_attribute> = <table_name2.join_attribut>

    AND <search_condition>

CAR	CARSALES	CUSTOMER
CarID	CarID	CustomerID
Make	CustomerID	CName
Model	SalesDate	CPhone
Year	PurchasedPrice	Address
Colour	StampDuty	
advertisedPrice		
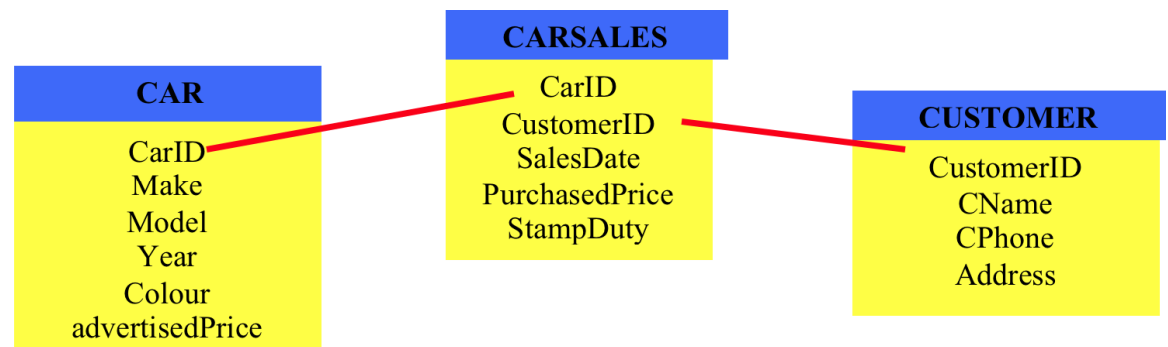
Joining Multiple Tables

- joining the CAR, CARSALES and CUSTOMER tables:

  SELECT customer.cname, customer.cphone, carsales.salesdate, carsales.purchasedPrice,
  carsales.stampDuty, car.make, car.model
  FROM customer, carsales, car
  WHERE customer.customerID = carsales.customerID
  AND   car.carID = carsales.carID;

- using prefix when joining tables:

  - when more than one table is involved, a prefix for each attribute is recommended to avoid ambiguity

  SELECT ct.cname, ct.cphone, cs.salesdate,
  cs.purchasedPrice,cs.stampDuty,
  c.make, c.model
  FROM customer ct, carsales cs, car c
  WHERE ct.customerID = cs.customerID
  AND   c.carID = cs.carID;;

  - **The output of both example:**

```
CNAME                CPHONE      SALESDATE    PURCHASEDPRICE  STAMPDUTY MAKE                    MODEL
-------------------- ---------- ----------   -------------- ---------- ----------------------  ----------
Leonard              0434251160 18-MAY-2016          98800     5631.6 BMW                      520d
Molly                0412908876 17-JUN-2016          98800     5631.6 BMW                      520d
Ben                  0433521126 18-MAY-2016          68200     3887.4 Audi                     A5
Lily                 0470457441 13-JUL-2016          12650        506 Holden                   Commodore
Jone                 0471257980 14-JUN-2016          13488     539.52 Honda                    Civic
Rex                  0456231879 12-DEC-2015          13999     559.96 Mazda                    3Series
Tomas                0498896110 14-JUN-2016          39888    1276.42 Volkwagen                Golf
Tomas                0498896110 13-JUL-2016           7490      299.6 Honda                    City
Tina                 0498698221 13-JUL-2016          23888     955.52 Toyota                   Corolla
Bill                 0423784435 04-FEB-2015          25780     824.96 Holden                   Cruze
```

Joining Multiple Tables

- ## Subquery

  - **Example:** *retrieving the make, model and advertisedPrice of the cars that are not sold*

    SELECT make, model, advertisedPrice FROM car
    WHERE carid **NOT IN**
       (SELECT carid FROM carsales);

    ```
    MAKE              MODEL             ADVERTISEDPRICE
    ----------------- ----------------- ---------------
    Subaru            Outback           33950
    Lexus             RX350             33500
    Mini              Cooper            19750
    Nissan            Dualis            18888
    Mercedes Benz     C200              16995
    Jeep              Grand Cherokee    21999
    6 rows selected
    ```

H. Aggregate Functions and Group By

- **Aggregate Functions**

  - summarize the input table

  - often used include:

    - COUNT - count number of records in the input table

    - SUM - calculate the sum of a numerical attribute

    - MIN and MAX - find the smallest and the largest value of a certain attribute

Aggregate Functions

- **Count(*)**

- **Example:** *returning the number of records that is available from the CAR table*

SELECT COUNT(*)

FROM car;

```
COUNT(*)
------------------
15
```

Aggregate Functions

COUNT(attribute) & COUNT(DISTINCT attribute)

Output:

- **Example:** *returning the number of values of colour that is available from the CAR table*

```
COLOUR
----------
Black
Grey
Black
Grey
Red
Black
Red
Blue
White
White
Blue
null
Grey
Black
White

15 rows selected
```

*SELECT **COUNT(colour)** FROM car;*

15 records include one NULL value

```
COUNT(COLOUR)

-------------

14
```

Cause by a record with null value

- **Example:** *returning the number of distinct values of colour that is available from the CAR table*

*SELECT **COUNT(DISTINCT colour)** FROM car;*

```
COUNT(DISTINCTCOLOUR)

----------------------

5
```

Aggregate Functions

- ***MIN and MAX***

- **Example:** *calculate the minimum and maximum advertisedPrice of the cars*

*SELECT **MIN**(advertisedPrice) AS "Lowest Price", **MAX**(advertisedPrice) AS "Highest Price"*

FROM car;

```
Lowest Price Highest Price
------------ -------------
        7490         98800
```

Aggregate Functions and Group By

- ## Group By

  - group an input table into a number of groups based on one or more nominated attributes

  - often used in conjunction with aggregate functions

    - *CAR table: group by Make*

      SELECT make, count(*)

      FROM car

      GROUP BY *make;*

MAKE	COUNT(*)
Holden	2
Lexus	1
Subaru	1
BMW	1
Mazda	1
Nissan	1
Audi	1
Volkwagen	1
Toyota	1
Honda	2
Mercedes Benz	1
Jeep	1
Mini	1

59

Aggregate Functions and Group By

- *CAR table: grouping by Make with combination of getting only the record groups that contain the count value greater than 1 is as follows*

  SELECT make, COUNT(DISTINCT model)

  FROM car

  GROUP BY make

  HAVING COUNT(DISTINCT model)>1;

```
MAKE                      COUNT(DISTINCTMODEL)

-------------------- --------------------

Holden                                     2

Honda                                      2
```

Aggregate Functions and Group By

- *multiple tables: selecting the number of sold cars for each colour*

  SELECT colour, COUNT(DINSTINCT customerId)

  FROM car c, carsales cs

  WHERE c.carid=cs.carid

  GROUP BY colour;

```
COLOUR          COUNT(DISTINCTCUSTOMERID)

-----------     -------------------------

White                                   2

Grey                                    4

Blue                                    1

Black                                   2

Red                                     1
```

Aggregate Functions

- ## Count vs. Sum

  - COUNT - count number of records in the input table

  - SUM - calculate the sum of a numerical attribute

  *SELECT <attribute1, attribute2>, **COUNT(*)***
  FROM <table_name>
  GROUP BY <attribute1, attribute2>;

  *SELECT <attribute1, attribute2>, **SUM (attribute3)***
  FROM <table_name>
  GROUP BY <attribute1, attribute2>;

Aggregate Functions

- ## Sum without Group By

  - **Example**: calculate the sum of purchasedPrice

    *SELECT **SUM**(purchasedPrice) AS totalsales*

    FROM carsales;

    ```
    TOTALSALES

    ----------

       402983
    ```

Aggregate Functions

- ## Sum with Group By

  - **Example**: calculate the sum of purchasedPrice for each make in
    carsales table

    *SELECT c.make, **SUM**(cs.purchasedPrice) AS totalsales*
    FROM carsales cs, car c
    WHERE c.CARID=cs.CARID
    GROUP BY c.make;

    ```
    MAKE                    TOTALSALES
    --------------------- ----------
    Holden                      38430
    BMW                        197600
    Mazda                       13999
    Audi                        68200
    ```

Simple Query Retrieval

- **SUM with Calculations**

  - **Example:** list all Cname, SalesDate and calculate the TotalPrice for each customer (TotalPrice = PurchasedPrice+StampDuty)

    *SELECT ct.cname, cs.salesdate, **SUM(purchasedPrice+StampDuty)** as TotalPrice*

    FROM carSales cs, customer ct

    WHERE cs.customerid=ct.customerid

    GROUP BY ct.cname, cs.salesdate;

    ```
    CNAME                 SALESDATE    TOTALPRICE

    --------------------- -----------  ----------

    Jone                  14-JUN-2016    14027.52

    Rex                   12-DEC-2015    14558.96

    Lily                  13-JUL-2016       13156
    ```

I. Alter Tables – Add New Fields

- **Syntax:**

  ALTER TABLE <table_name>

  ADD (attribute_name data_type_declare constraints_declare);

  - *attribute name*: referring to the new attribute that you want to add into the existing table

  - *data type*: defines the data type and the size of the new attribute

  - *constraint*: defines the constraints that the new attribute might be enforced by certain constraints

- **Example:** *add a transmission attribute to the CAR table*

  ALTER TABLE car
  ADD (transmission VARCHAR2(10));

Alter Tables – Modify Fields

- **Syntax:**

    ALTER TABLE <table_name>

    MODIFY (attribute_name new_data_type);

    - *attribute name:* refers to the attribute that you want to modify

    - *new data:* defines the new data type that you want to use replacing the old one

- **Example:** *change the data type of the transmission to CHAR with a size of 30*

    ALTER TABLE car

    MODIFY (transmission CHAR(30));

Alter Tables – Drop Columns

- **Syntax:**

  ALTER TABLE <table_name>

  DROP COLUMN attribute_name;

- **Example:** *delete the attribute transmission from the CAR table*

  ALTER TABLE car

  DROP COLUMN transmission ;