# Simple select queries

## Simple selecting query

Do a simple select query on the **Fnd\_Event\_tab** and display all of its columns and all rows.

SELECT \*

FROM Fnd\_Event\_tab

1.2. Selecting by some columns

Display columns Event\_Lu\_name, Event\_ID and Event\_Description for all records from the **Fnd\_Event\_tab** .

SELECT Event\_Lu\_name, Event\_ID,Event\_Description

FROM **Fnd\_Event\_tab**

## Query with condition

1. Display records from the **fnd\_event\_parameter\_tab** where   
   Event\_LU\_name = ‘FndMonitorEntry’

SELECT \*

FROM fnd\_event\_parameter\_tab

WHERE Event\_LU\_name = ‘FndMonitorEntry’

1. Display records from the **all\_indexes** where

INITIAL\_EXTENT > 16384

SELECT \*

FROM all\_indexes

WHERE INITIAL\_EXTENT > 16384

## Query with pattern matching

Display Event\_lu\_name of all records from **fnd\_event\_tab** where Event\_idends with ‘ED’.

SELECT Event\_lu\_name

FROM fnd\_event\_ tab

WHERE Event\_id LIKE ‘%eED’

## How do you get a description of a table?

Display a description like the one below for the **fnd\_user\_tab** table.

Name Type Nullable Default Id Comments

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

IDENTITY VARCHAR2(30) NOT NULL 1

DESCRIPTION VARCHAR2(2000) NOT NULL 2

ORACLE\_USER VARCHAR2(30) 3

WEB\_USER VARCHAR2(2000) 4

ACTIVE VARCHAR2(5) NOT NULL 5

TEXT\_ID$ VARCHAR2(50) NOT NULL 6

VALID\_FROM DATE 7

VALID\_TO DATE 8

ROWVERSION NUMBER NOT NULL 9

ROWKEY VARCHAR2(50) 10

DESC fnd\_user\_tab

## Selecting with several conditions (combination of logic)

Display all records from **all\_indexes** where owner = SYS OR IFSAPP and initial\_extent = 65536

SELECT \* FROM all\_indexes WHERE (owner = 'SYS' OR owner = 'IFSAPP') AND initial\_extent = 65536Another logical query

Display all records from **all\_indexes** where uniqueness = ‘UNIQUE’ and tablespace\_name =’SYSTEM’

## A simple calculation

Calculate the product of 23 and 14. The result should look like the following.

23\*14

322

SELECT 23\*14 FROM DUAL;

## Query by ordering

Display all records from **all\_indexes** where uniqueness = ‘UNIQUE’ AND initial\_extent = 65536

Order it by Owner Ascending order.

SELECT \* FROM all\_indexes WHERE uniqueness = ‘UNIQUE’ AND initial\_extent = 65536

## Distinct rows and pattern matching

Select distinct event\_lu\_name from **fnd\_event\_parameter\_tab**

SELECT DISTINCT event\_lu\_name FROM fnd\_event\_parameter\_tab

## Using concatenation for layout

Write a query so that you get the result like below from **Fnd\_Event\_tab.**

"LU name Description"  
 ----------------------------------------------------------------------

DataArchiveUtil Data Archive executed

FndMonitorEntry An executed monitor entry has created a warning

HistoryLogAttribute History log modified

Transaction Background job processed by batch handler

Transaction Background job in progress and partly ready

FndLog State of Project Log entry changed

PrintJob Creation of a report as pdf-file is completed

ReplicationLog Replication error registered

Security Security Checkpoint success

General Concurrent Users Warning

WorkTimeCalendar A calendar has been generated

SELECT CONCAT(CONCAT(event\_lu\_name, ' '), event\_description) AS “LU name Description” FROM Fnd\_Event\_tab

# Queries with functions

## Selecting by character function

Display LU names and their description for all entries from the **fnd\_event\_tab** table.

The result should look like the following.

LU NAME DESCRIPTION

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

Dataarchiveutil Data Archive Executed

Fndmonitorentry An Executed Monitor Entry Has Created A Warning

Historylogattribute History Log Modified

Transaction Background Job Processed By Batch Handler

Transaction Background Job In Progress And Partly Ready

Fndlog State Of Project Log Entry Changed

Printjob Creation Of A Report As Pdf-File Is Completed

Replicationlog Replication Error Registered

Security Security Checkpoint Success

General Concurrent Users Warning

Worktimecalendar A Calendar Has Been Generated

SELECT event\_lu\_name AS "LU NAME", event\_description AS "DESCRIPTION" FROM Fnd\_Event\_tab

## Selecting by character function

Generate the output below from **fnd\_event\_tab**.

RPAD(T.EVENT\_LU\_NAME,30,' ') LPAD(T.EVENT\_DESCRIPTION,50,'\*’)

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

DataArchiveUtil \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Data Archive executed

FndMonitorEntry \*\*\*An executed monitor entry has created a warning

HistoryLogAttribute \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*History log modified

Transaction \*\*\*\*\*\*\*\*\*Background job processed by batch handler

Transaction \*\*\*\*\*\*\*Background job in progress and partly ready

FndLog \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*State of Project Log entry changed

PrintJob \*\*\*\*\*Creation of a report as pdf-file is completed

ReplicationLog \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Replication error registered

Security \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Security Checkpoint success

General \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Concurrent Users Warning

IntfaceEvents \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Data Migration Job Finished

IntfaceEvents \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Data Migration Error Job

WorkTimeCalendar \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*A calendar has been generated

DocDistEngine \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*User has received a new document

ApprovalRouting \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Approval Route is completed

ApprovalRouting \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Approval Route is rejected

ApprovalRouting \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*New step is ready for approval

BatchTransferHandler \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Document Transfer is Complete

EdmFileOpAnnounce \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*User read the document.

EdmFileOpAnnounce \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*User modified the document.

SELECT RPAD(T.EVENT\_LU\_NAME,30,' '), LPAD(T.EVENT\_DESCRIPTION,50,'\*')

FROM Fnd\_Event\_tab T

## 2.3. Selecting by character function

Generate the output below from **fnd\_event\_tab.**

EVENT\_LU\_NAME LENGTH(T.EVENT\_LU\_NAME)

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

Activity 8

Activity 8

Activity 8

Activity 8

Activity 8

Activity 8

Activity 8

ApprovalRouting 15

ApprovalRouting 15

ApprovalRouting 15

BatchTransferHandler 20

CompAvailable 13

CreditCollectionInfo 20

CustCreditInfoUtil 18

CustCreditInfoUtil 18

CustomerCreditInfo 18

CustomerCreditNote 18

CustomerOrder 13

CustomerOrder 13

SELECT event\_lu\_name, LENGTH(event\_lu\_name) AS "DESCRIPTION" FROM Fnd\_Event\_tab

## 2.4. Selecting by number function

Display the index names and their rows numbers where the NUM\_ROWS is higher than 1000, from the **SYS.ALL\_INDEXES**. The figures in the NUM\_ROWS column should be displayed in blocks of 1000.

INDEX\_NAME (1000 Rows)

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

WORK\_TIME\_COUNTER\_DESC\_3\_IX 10

WORK\_TIME\_COUNTER\_DESC\_2\_IX 10

TERM\_DISPLAY\_NAME\_IX2 49

TERM\_DISPLAY\_NAME\_IX1 49

FNDRR\_CLIENT\_PROFILE\_VALUE\_PK 2

PRINT\_JOB\_IX 1

PRINT\_JOB\_PK 1

LOGICAL\_UNIT\_PK 2

TERM\_USAGE\_DEFINITION\_UK 37

TERM\_USAGE\_DEFINITION\_PK 37

TERM\_USAGE\_DEFINITION\_IX2 37

TERM\_USAGE\_DEFINITION\_IX1 37

TERM\_TRANSLATED\_NAME\_PK 28

TERM\_TRANSLATED\_DEFINITION\_PK 24

TERM\_UK 23

TERM\_TAB\_IX1 23

TERM\_PK 23

FNDDR\_VIEW\_ENT\_FK 1

FNDDR\_VIEW\_BASED\_ON\_FK 1

FNDDR\_VIEW\_ALT\_IX 1

SELECT index\_name, ROUND(num\_rows/1000) AS "(1000 Rows)"

FROM all\_indexes

WHERE num\_rows > 1000

## Selecting by number function

Display INDEX\_NAME, NUM\_ROWS, a column with numbers from –1 to 1, indicating if the NUM\_ROWS is less than 1000 or more than 1000 from the **all\_indexes** view.

INDEX\_NAME NUM\_ROWS INDICATOR

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

TMP\_RESULT\_PK 38 -1

NPP\_TASK\_DETAIL\_PK 10 -1

NAME\_PK 13 -1

NPP\_TASK\_RANK\_PK 13 -1

WORK\_TIME\_COUNTER\_DESC\_3\_IX 10758 1

WORK\_TIME\_COUNTER\_DESC\_2\_IX 10758 1

TERM\_DISPLAY\_NAME\_IX2 49369 1

TERM\_DISPLAY\_NAME\_IX1 49369 1

SYS\_C0017255 1 -1

SYS\_C0017201 2 -1

NPP\_ACCOUNTING\_PARAM\_PK 18 -1

NPP\_PARAM\_PK 4 -1

NPP\_PURCHASE\_FORECAST\_PK 0 -1

NPP\_OMIT\_VOUCHER\_ROW\_PK 0 -1

NPP\_OMIT\_VOUCHER\_PK 0 -1

NPP\_HIST\_VOUCHER\_ROW\_PK 0 -1

NPP\_HIST\_VOUCHER\_PK 0 -1

STANDARD\_MFG\_PROCESS\_PK 0 -1

NPP\_SUBBRANCH\_OFFICE\_PK 2 -1

NPP\_SETTLEMENT\_ 2 -1

SELECT index\_name, num\_rows, DECODE(SIGN(num\_rows-1000),-1,'-1',+1,'1') AS “INDICATOR”

FROM all\_indexes

## Selecting by date function

Display every INDEX\_NAME with the LAST\_ANALYZED and also calculate the Following Fridays date from the **all\_indexes** view.

INDEX\_NAME LAST\_ANALYZED The Next Friday

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

INVENTORY\_VALUE\_PK 2008-03-19 5: 2008-03-21 5:04

INVENTORY\_VALUE\_SIMULATION\_PK 2008-03-19 5: 2008-03-21 5:04

INVENTORY\_VALUE\_SIM\_LINE\_PK 2008-03-19 5: 2008-03-21 5:04

INVENT\_VALUE\_PART\_DETAIL\_PK 2008-03-19 5: 2008-03-21 5:04

INVENTORY\_VALUE\_PART\_PK 2008-03-19 5: 2008-03-21 5:04

INVENTORY\_TRANSACTION\_HIST\_PK 2008-03-19 5: 2008-03-21 5:04

INVENTORY\_TRANSACTION\_HIS10\_IX 2008-03-19 5: 2008-03-21 5:04

INVENTORY\_TRANSACTION\_HIST9\_IX 2008-03-19 5: 2008-03-21 5:04

INVENTORY\_TRANSACTION\_HIST6\_IX 2008-03-19 5: 2008-03-21 5:04

INVENTORY\_TRANSACTION\_HIST4\_IX 2008-03-19 5: 2008-03-21 5:04

INVENTORY\_TRANSACTION\_HIST3\_IX 2008-03-19 5: 2008-03-21 5:04

INVENTORY\_TRANSACTION\_HIST2\_IX 2008-03-19 5: 2008-03-21 5:04

INVENTORY\_TRANSACTION\_HIST1\_IX 2008-03-19 5: 2008-03-21 5:04

INVENTORY\_TRANSACTION\_HIS12\_IX 2008-03-19 5: 2008-03-21 5:04

NPP\_INV\_TRAN\_HIST\_IX02 2008-03-19 5: 2008-03-21 5:04

NPP\_INV\_TRAN\_HIST\_IX01 2008-03-19 5: 2008-03-21 5:04

INVENTORY\_PRODUCT\_FAMILY\_PK 2008-03-19 5: 2008-03-21 5:04

INVENTORY\_PRODUCT\_CODE\_PK 2008-03-19 5: 2008-03-21 5:04

INV\_PROD\_CODE\_COST\_SOURCE\_PK 2008-03-19 5: 2008-03-21 5:04

INVENTORY\_PART\_PK 2008-03-19 5: 2008-03-21 5:04

INVENTORY\_PART\_1\_IX 2008-03-19 5: 2008-03-21 5:04

SELECT index\_name, LAST\_ANALYZED,

NEXT\_DAY(LAST\_ANALYZED,'FRIDAY') AS "NEXT FRIDAY"

FROM all\_indexes

WHERE num\_rows > 1000

## Selecting by conversion function

Modify the previous question to have an output result that should look like the following.

*(Hint: Nest several functions!)*

INDEX\_NAME T.LAST\_ANALYZED The Next Friday

CUSTOMER\_ORDER\_LINE\_4\_IX MARCH - THURSDAY -08 APRIL -FRIDAY -08

(Month – Day – Year)

SELECT index\_name,

TO\_CHAR(TO\_DATE(last\_analyzed, 'yyyy-mm-dd hh:mi:ssam'), 'MONTH -DAY -YY') AS "LAST\_ANALYZED",

TO\_CHAR(TO\_DATE(NEXT\_DAY(last\_analyzed,'FRIDAY'), 'yyyy-mm-dd hh:mi:ssam'), 'MONTH - DAY - YY') AS "NEXT FRIDAY"

FROM all\_indexes

WHERE num\_rows > 1000

## Selecting by transformation function

Display the INDEX\_NAME and the NUM\_ROWS column and then state if they have more than 1000 records depending on the NUM\_ROWS column from the **all\_indexes** view.

*(Hint: Nest several functions!)*

If NUM\_ROWS less than 1000 then ‘few rows’

If NUM\_ROWS more than 1000 then ‘more than 1000’

If NUM\_ROWS equal 1000 then '1000 rows'

INDEX\_NAME NUM\_ROWS Comment on rows

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

I\_OBJ3 2939 more than 1000

I\_OBJ1 281386 more than 1000

I\_COL2 819406 more than 1000

I\_OBJ2 281386 more than 1000

I\_COL3 819406 more than 1000

I\_COL1 820219 more than 1000

I\_COM1 361619 more than 1000

I\_ARGUMENT1 1713562 more than 1000

I\_ARGUMENT2 1915956 more than 1000

I\_SOURCE1 30335120 more than 1000

I\_PENDING\_TRANS1 0 few rows

I\_SYSTEM\_PRIVILEGE\_MAP 166 few rows

I\_TABLE\_PRIVILEGE\_MAP 24 few rows

I\_STMT\_AUDIT\_OPTION\_MAP 205 few rows

I\_AUD1 0 few rows

I\_AUDIT\_ACTIONS 160 few rows

OL$NAME

OL$SIGNATURE

OL$HNT\_NUM

IMPDP\_STATS

SELECT index\_name, num\_rows, DECODE(SIGN(num\_rows-1000),-1,'FEW ROWS',0,'1000 ROWS',+1,'MORE THAN 1000') AS "Comment on rows"

FROM all\_indexes

## Selecting by combining several functions\*

Display the INDEX\_NAME and the INDEX\_TYPE and an extra column stating if it is an ‘ORACLE\_INDEX’ for the records in the **all\_indexes** view.

Decide if the record is an oracle index by checking if the INDEX\_NAME starts with SYS%. Consider records which have num\_rows < 1000 and distinct\_keys between 0 and 500 only.

INDEX\_NAME INDEX\_TYPE NEW NAME

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

SYS\_C003149 NORMAL ORACLE\_INDEX

SYS\_C003150 NORMAL ORACLE\_INDEX

SYS\_C003151 NORMAL ORACLE\_INDEX

SYS\_C003152 NORMAL ORACLE\_INDEX

SYS\_C003154 NORMAL ORACLE\_INDEX

SYS\_C003155 NORMAL ORACLE\_INDEX

SYS\_C003156 NORMAL ORACLE\_INDEX

SYS\_C003157 NORMAL ORACLE\_INDEX

SYS\_C003158 NORMAL ORACLE\_INDEX

SYS\_C003159 NORMAL ORACLE\_INDEX

SYS\_C003160 NORMAL ORACLE\_INDEX

SYS\_C003161 NORMAL ORACLE\_INDEX

XDB$SCHEMA\_URL NORMAL XDB

SYS\_C003164 NORMAL ORACLE\_INDEX

PK\_SRID NORMAL PK\_

SYS\_C003283 NORMAL ORACLE\_INDEX

UNIT\_OF\_MEASURE\_PRIM NORMAL UNI

SELECT index\_name, index\_type, DECODE(SUBSTR(index\_name,1,3),'SYS','ORACLE\_INDEX',SUBSTR(index\_name,1,3)) "New Name"

FROM all\_indexes

WHERE num\_rows < 1000 AND distinct\_keys BETWEEN 0 AND 500

## Selecting combining several functions\*

Calculate the average num\_rows the for all indexes belonging to a certain table owner from the **all\_indexes** view.

TABLE\_OWNER AVG

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

MDSYS 1104.7586206896

CTXSYS 186

OLAPSYS 0

SYSTEM 978

EXFSYS

IFS01 145.77947295423

XDB 220.33333333333

SYS 1334005.4285714

WMSYS 0.5454545454545

SELECT table\_owner, AVG(num\_rows)

FROM all\_indexes

GROUP BY table\_owner

## Selecting by combining several functions\*

Display INDEX\_NAME, LAST\_ANALYZED and DAYS\_AGO by having the difference between the last analyzed date and the current date from the **all\_indexes** view.

INDEX\_NAME LAST\_ANALYZED DAYS AGO

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

I\_OBJ3 5/5/2008 7

I\_OBJ1 5/5/2008 7

I\_COL2 4/29/2008 13

I\_OBJ2 5/5/2008 7

I\_COL3 4/29/2008 13

I\_COL1 4/29/2008 13

I\_COM1 4/29/2008 13

I\_ARGUMENT1 5/5/2008 7

I\_ARGUMENT2 5/5/2008 7

I\_SOURCE1 4/29/2008 13

SELECT index\_name, last\_analyzed, TRUNC(sysdate) - TO\_DATE(last\_analyzed, 'yyyy-mm-dd hh:mi:ssam')

FROM all\_indexes

# Joining and grouping queries

## Joining two tables

Display event\_lu\_name, event\_id, event\_description, id, id\_type from the

**fnd\_event\_tab** and **fnd\_event\_parameter\_tab** .

SELECT et.event\_lu\_name, et.event\_id, et.event\_description, ept.id, ept.id\_type

FROM fnd\_event\_tab et INNER JOIN fnd\_event\_parameter\_tab ept

ON et.event\_lu\_name = ept.event\_lu\_name

## Joining, selecting rows from a list of values

Display the above query but only records with the parameter’s row version that are within the month of May 2016.

SELECT et.event\_lu\_name, et.event\_id, et.event\_description, ept.id, ept.id\_type

FROM fnd\_event\_tab et INNER JOIN fnd\_event\_parameter\_tab ept

ON et.event\_lu\_name = ept.event\_lu\_name

WHERE EXTRACT(month FROM ept.rowversion) = 5 AND EXTRACT(year FROM ept.rowversion) = 2016

## Join a table by itself

Display those records in the **fnd\_event\_parameter\_tab** that have the same current\_value and new\_value.

SELECT \*

FROM fnd\_event\_parameter\_tab ept1, fnd\_event\_parameter\_tab ept2

WHERE ept1.current\_value = ept2.new\_value

## Outer join

### Display *event\_lu\_name,event\_description* from fnd\_event\_tab and event\_id,action\_number from fnd\_event\_action\_tab using an outer join.

SELECT et.event\_lu\_name, et.event\_id, et.event\_description, ept.id, ept.id\_type

FROM fnd\_event\_tab et FULL OUTER JOIN fnd\_event\_parameter\_tab ept

ON et.event\_lu\_name = ept.event\_lu\_name

## Grouping data together from several tables

Display the count of parameter in the **fnd\_event\_parameter\_tab** for each event\_id and event\_lu\_name along with the event\_description.

SELECT et.event\_id, COUNT(et.event\_id), et.event\_description

FROM fnd\_event\_tab et JOIN fnd\_event\_parameter\_tab ept ON et.event\_lu\_name = ept.event\_lu\_name AND et.event\_id=ept.event\_id

GROUP BY et.event\_id, et.event\_lu\_name, et.event\_description

## Selecting by group criteria\*

Display the count of parameter in the **fnd\_event\_parameter\_tab** for each event\_id and event\_lu\_name along with the event\_description but only event\_id and event\_lu\_name that have more than 10 parameters.

SELECT et.event\_id, COUNT(et.event\_id), et.event\_description

FROM fnd\_event\_tab et JOIN fnd\_event\_parameter\_tab ept ON et.event\_lu\_name = ept.event\_lu\_name AND et.event\_id=ept.event\_id

GROUP BY et.event\_id, et.event\_lu\_name, et.event\_description

HAVING COUNT(et.event\_id)>10

## Grouping data together

Display the Min and Max counts for event\_id and event\_lu\_name in the **fnd\_event\_parameter\_tab.**

## Selecting by a sub query

Display only the event\_id and event\_lu\_name and its event\_description that has the maximum parameter count using the **fnd\_event\_parameter\_tab** and **fnd\_event\_tab**.

SELECT et.event\_id, COUNT(et.event\_id), et.event\_description

FROM fnd\_event\_tab et JOIN fnd\_event\_parameter\_tab ept ON et.event\_lu\_name = ept.event\_lu\_name AND et.event\_id=ept.event\_id

GROUP BY et.event\_id, et.event\_lu\_name, et.event\_description

ORDER BY COUNT(et.event\_id) DESC

# Data Definition Language (DDL) and Data Manipulation Language (DML)

## Create a new table

Create a new table with your own definition. For example, a ‘Friends’ table with name, address, phone etc. for your friends. Prefix the table name with ‘demo\_’ and do not forget to state if the contents of the columns are allowed to be empty or not.

To get a maintainable table, you may do not create more than at most seven columns.

CREATE TABLE demo\_student

(

st\_id NUMBER(10) NOT NULL,

name VARCHAR2(100) NOT NULL,

class VARCHAR2(5) NOT NULL,

dob DATE NOT NULL

CONSTRAINT demo\_student\_pk PRIMARY KEY (st\_id)

);

## Insert some records

Insert some records to your created table.

INSERT INTO demo\_student VALUES( 1, ‘Peter Hale’, ‘12M-Si-1’, 1999-10-20);

INSERT INTO demo\_student VALUES( 2, ‘Eric Selvig, ‘12M-Si-2’, 1999-11-30);

INSERT INTO demo\_student VALUES( 3, ‘Kurt Smith, ‘12A-Si-1’, 1999-02-26);

INSERT INTO demo\_student VALUES( 4, ‘Tom Riddle’, ‘13M-En-1’, 1998-08-15);

## Update table and/or records

Did you discover that you have forgotten something? Maybe you want to add a column or just want to modify some rows that fit some special condition. Even if you don’t feel that way, try to do some updates, to see how it works.

UPDATE demo\_student SET name=‘Will Smith’, class=‘13M-En-2’, dob=1998-07-15 WHERE st\_id=3;

## Delete records

To maintain your table, delete those records that you do not want to exist in your table any more.

DELETE FROM demo\_student WHERE st\_id=3;

## Drop table

Before we end these exercises, get rid of the tables you created from the database.

DROP TABLE demo\_student PURGE3;