SQA – Testing Key Concepts:

1. Quality Objectives: Define clear and measurable quality objectives aligned with project goals. Specify quality metrics and acceptance criteria.
2. Test Planning: Develop a comprehensive test plan: Test strategy, scope, and objectives. Resource allocation, timelines, and budgets. Risk assessment and mitigation strategies.
3. Collaboration and Communication: Foster collaboration between development, testing, and other stakeholders. Use agile methodologies and regular meetings for effective communication.

4. **Testing Types:**

- **Functional**: **API / UI** unit / integration / system testing – and acceptance.
- **Non-functional**: performance, security, usability, compatibility & scalability. *Unlike Functional which addresses the Product Functionality, These Testing's check the Over-All Characteristics & Compliance to Quality Standards in General (X-ability).*
- **Regression**: continuously verify existing functionalities (using automation).
- **Integration**: validate interactions of product components to external systems.
- **Load & Stress**: assess system behavior under heavy load & stress conditions.
- **Automated**: develop robust test automation frameworks for efficiency.
- **Exploratory**: combine structured test cases with exploratory techniques.
- **Security**: identify and mitigate security vulnerabilities.
- **Data**: verify data accuracy, integrity, and consistency.

5. Test Artifacts: Create and maintain detailed test cases, test data, and test scripts. Document traceability to requirements and defects.
6. Test Execution and Management: Execute test cases, record results, and manage test environments. Prioritize test execution based on risk and criticality. Use test management tools for test case management and reporting.
7. Defect Management: Report and track defects through their lifecycle. Implement a defect triage process for prioritization. Validate defect fixes and perform root cause analysis.
8. Metrics and Reporting: Collect and analyze quality metrics: Test coverage, code coverage, defect density, and test pass rates. Generate comprehensive test reports for stakeholders. Implement test dashboards for real-time visibility.
9. Continuous Improvement: Conduct retrospectives and lessons learned sessions. Implement process improvements based on feedback and data. Promote a culture of continuous learning and innovation.

Tools and Technologies:

1. Test Management Tools: JIRA, TestRail, ALM, or custom tools for test management.
2. Version Control: Git for versioning test scripts and automation code.
3. Automation Tools: Selenium, Appium, JUnit, TestNG, or specialized automation tools.
4. Metric Analysis Tools: Grafana, Kibana, or custom dashboards for tracking metrics.

Common Pitfalls:

1. Late Testing: Start testing early in the SDLC.
2. Neglecting Non-functional Testing: Include non-functional testing in Test Strategy.
3. Incomplete Test Documentation: Maintain clear (K.I.S.S), comprehensive & up-to-date test documentation – easy to be located & accessed.
4. Ignoring Defects: Prioritize and address defects promptly.
5. Rigidity: Be flexible and adaptable to changing requirements.

Tips for Success:

1. Advocate Quality: Make quality a shared responsibility.
2. Continuous Learning: Stay updated with emerging technology, best practices.
3. Innovation: Encourage creative testing approaches and tools.
4. Documentation Excellence: Maintain detailed and organized records.

Remember, advanced SQA Engineering is crucial for delivering high-quality software that meets user expectations and business objectives. Use this cheat sheet to guide your SQA processes and continually refine your approach.

Basic Fundamental Concepts in **Quality Assurance** (QA):

1. Quality Assurance: refers to the process of ensuring that product or service meets specified requirements and standards. It involves planning, designing, implementing, and executing various activities to prevent defects and improve overall quality.
2. Testing: process of evaluating a system or component to identify any differences between expected and actual results. It involves executing test cases, analyzing outcomes, and reporting defects.
3. Test Planning: involves defining the scope, objectives, and approach for testing & includes identifying test environment, test resources, and test deliverables. Test planning ensures that testing is well-organized and aligns with project goals.
4. Test Design: involves creating test cases and test scenarios based on requirements or specifications. It focuses on identifying test conditions, test data, and expected results. Effective test design helps achieve maximum test coverage and identifies defects efficiently.
5. Test Execution: the process of running test cases and capturing the actual results. It involves comparing the actual outcomes with the expected results and reporting any discrepancies or defects.
6. Defect Management: involves identifying, logging, prioritizing, and tracking software defects. It includes defect triage, where defects are reviewed and assigned to appropriate teams for resolution.
7. Test Automation: involves using software tools and scripts to automate repetitive and manual testing tasks. It helps improve efficiency, reduces human errors, and enables faster testing cycles.
8. Test Metrics and Reporting: quantitative measures used to assess the quality and progress of testing activities. Metrics provide insights into test coverage, defect density, test execution status, and other key indicators. Test reports summarize the test results and provide stakeholders with an overview of the testing effort.
9. Continuous Integration & Delivery (CI/CD): an approach that involves integrating code changes frequently and delivering software in an automated and continuous manner. It ensures that changes are tested early and often, reducing the risk of defects.
10. Risk-based Testing: involves identifying and prioritizing test efforts based on the potential impact and likelihood of failures. It helps allocate testing resources effectively and focuses on high-risk areas.

» **Verification** › Did I build the Product right (to meet given requirements)?
» **Validation** › Did I build the right Product (to meet the customers' needs)?

BUGS:

» After finding a problem - Filling a Bug:
› Title: Short Description = Login with wrong password give wrong error
› Pre-Conditions: Before you Start = Login page open
› Scenario: What you do = Enter in proper place the Username with wrong Password and click to Login
› Expected results: What should happen = Login failed with Error "wrong password"
› Actual result: What happened = Login failed with Error "please try again"
› Attached: Logs, Print-screen or any other thing.

```
Test Type, Method and Technique:
--------------------------------
- Positive: Test a specific requirement in its nominal values (situations)
- Negative: Test a specific requirement with non-valid values (inputs)
- Boundaries: Test a specific requirement with minimal/maximal values
- End cases: Test specific requirement under non-trivial\extreme condition
- Detailed verification
- End-to-end scenarios
- White box + Black box techniques
```

Test Types

Functional QA testing: Based on **Black Box** testing - verifying the functionality of a software application or system to ensure that it meets the specified requirements.

Non-Functional QA Assessments: Based on Standards (see: **Ability & END** of List).

Positive testing: Function as expected with valid inputs, under normal conditions !

Negative testing: How software handles invalid \ unexpected input \ error condition

Unit Testing: Verify functionality of individual units or components (functions, methods, or classes). Typically performed by developers during the development.

Integration Testing: Conducted to verify the interaction and data flow between different components, modules, or systems. The objective is to identify any issues that may arise when integrating individual components.

System Testing: Verify the complete & integrated software system to ensure that it meets the specified requirements. It involves testing the software as a whole and includes functionality, performance, security, and other aspects.

Acceptance Testing: Performed by Company testers & Client representatives to ensure that the software meets the requirements & is ready for deployment. Focuses on validating user's perspective & often involve real-world scenarios.

Delivery & Deploy Testing: Performed by QA in-charge of Installation process testing – Validate proper deploy of each component in all aspects… !!!

Functional Testing: Functional tests aim to verify that the software functions as expected and meets the specified functional requirements. It involves testing individual features, user interfaces, input validations, and error handling.

Regression Testing: Performed after making changes to the software, to ensure the existing functionality is not affected by modifications. Involves retesting previously tested features, to uncover any new defects or regressions.

Progression Testing: Ensure that new functionality added to software is as expected and meets the specified functional requirements. Involves testing ***individual features***, ***user interfaces***, ***input validations*** & ***error handling***.

Smoke Testing: Quick Test - performed to ensure that most critical features of an application function correctly. Typically executed after deploy before comprehensive testing - Acts as a preliminary check even Prior to Sanity.

Sanity Testing: Testing focused on quickly checking the core functionalities of application, to determine if it is stable enough for more even comprehensive testing. Purpose of sanity is to identify major issues early in the dev-life-cycle.

End-to-End Testing: Verify the entire software system, including multiple components, interfaces, and interactions between them. It aims to validate the flows and functionalities of the entire system as a user would experience it.

GUI Testing: Verify the visual aspects of the client-software (Graphical User Interface) components and interactions between the user & interface - aims to ensure a smooth and intuitive user experience & validate proper flows and functionality of all the interactive.

Layout and Design Testing: GUI testing assesses the visual layout, design consistency, and responsiveness of the user interface. It involves checking the placement, alignment, spacing, and size of UI elements to ensure they adhere to the design guidelines.

Error Handling Testing: Evaluate how software handles various error scenarios, such as invalid inputs, exceptions, or unexpected system behaviors. Ensures appropriate Error Messages displayed and system recovers gracefully.

Boundary & End Cases Testing: Check software properly validate & handles edge cases of data inputs at the boundaries of acceptable values (valid & invalid ranges at lower & upper limits, and values just outside those limits).

API Testing: Validates the functionality, reliability, and security of the software's APIs = Application Programming Interface. It involves testing API endpoints, request/response formats, authentication mechanisms, and data integrity.

Data correctness: Data all-over integrity in ***Storing the Data*** & ***Displaying it***.

Data Integrity Testing: Data integrity tests verify the accuracy, consistency, and integrity of data stored or processed by the software. It includes validating data inputs, outputs, calculations, and database interactions.

Data Conversions Testing: Verify that one data format can be converted into another data format so that the converted data format can be used seamlessly by the application under test appropriately.

Compatibility Testing: Verify software is compatible with different hardware, operating systems, browsers or devices. It ensures that the application works correctly in various network environments and configurations.

Usability Testing: Evaluate how user-friendly the software is by assessing its ease of use, intuitiveness, and overall user experience. Testers simulate user interactions and observe user behavior to identify any usability issues.

Accessibility (for disabled) Testing: Verify whether the software is accessible to individuals with **disabilities**. It ensures compliance with accessibility standards.

Localization Testing: Assess whether software has been properly adapted to support different languages, cultures, and locales. Checks accurate translations, correct date and time formats, currency symbols and localized elements.

Internationalization Testing: Focus on designing and developing software in a way that makes it easier to localize. It involves testing the software's ability to handle international data formats, character encodings & cultural differences.

Security Testing: Assess the software's resilience against potential vulnerabilities, threats, or unauthorized access. It includes testing authentication, authorization, encryption, input validation, and other security mechanisms.

Fault Tolerance, High Availability and Geo Redundancy Testing: Assess the software's ability to enable users' access & get proper \ limited functionality of system while one or more components fail !

Recovery Testing: Evaluate system's ability to recover from failures or disruptions (as varied crashes and loss of power or network), to ensure restart & resumes stable operate state & maintain data integrity (no data corruptions).

Back-Up & Restore Testing: Evaluate the effectiveness of an organization's software and methods of replicating data for security and its ability to reliably retrieve that data should the need arise…

Performance Testing: Evaluate the software's performance under expected or increased load conditions. It involves measuring response times, throughput, scalability, and resource usage to identify performance bottlenecks.

Load Testing: Assesses the software's performance under anticipated user loads. It simulates a high volume of concurrent users or transactions to determine how system handles the increased load and if it meets perf-criteria.

Stability and Durability Testing: Technique used to determine the characteristics of a system under various load conditions over time. This helps us to identify the stability of transaction response times over the duration of test.

Stress Testing: Evaluates software's performance under extreme conditions, such as high concurrent users, heavy data loads, or limited system resources. Aims to identify the breaking point or performance degradation of the system.

Compliance (to standards) Testing: Ensure the software meets regulatory, legal & industry-specific requirements. It involves verifying adherence to standards, regulations, data protection laws, and specific business rules.

Installation Testing: Check the software's installation process to ensure it can be installed, upgraded, or uninstalled correctly on various platforms. It includes verifying prerequisites, file integrity, and proper configuration. Installability…

Configuration Testing: Validates the software's behavior and functionality under different configuration settings. It verifies that the software works correctly with different settings, options, or parameter combinations. Configurability…

Interoperability Testing: Assess software's ability to interact and function correctly with other systems, platforms, or **external interfaces**. It validates seamless data exchange & compatibility with **third-party applications integrations**.

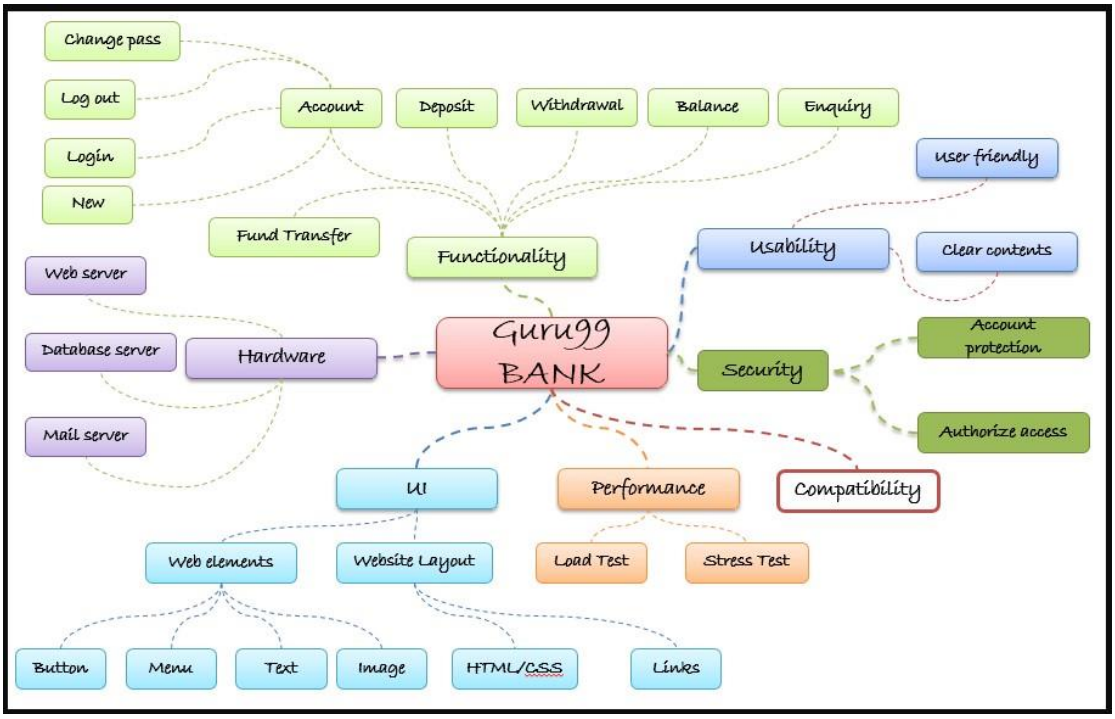
Efficiency: Assess systems' using Resources (memory, storage etc.) most efficiently.

Integrity: Assess the over-all resiliency to faults (power-out, heat, network, attack).

Reliability: Assess systems' ability to function properly over-time without issues.

More Non-Functional: **survivability, maintainability, portability, expendability, safety of usage**…

דוגמא:



באופן כללי:

אף אחד לא רוצה לספק מוצר קלוקל בכוונת תחילה – אבל לפעמים, ללא כוונה המוצר המסופק הוא קלוקל ! למה זה קורה ? נכון אפשר להאשים את כל שרשרת הניהול, אבל בסוף יש מישהו שהוא השער דרכו המוצר יוצא, ותפקידו הוא לעצור אותו מלצאת במידה והוא איננו עומד בסטנדרטים נאותים ! בדיקת המוצר לא תמנע ממנו מלהיות גרוע אם הוא כזה, אבל בדיקות גרועות תבטחנה שהוא יהיה כזה – והבדיקות לרוב הם מושא להתכנסות לחצים מכל הכיוונים !!! בדיקות מוצר באות להעריך עד כמה המוצר מתנהג לפי התכנון ומדלוור לפי ספסיפיקאציות (דרישות מוגדרות) – אבל כאשר הלחץ על שיחרור המוצר גובר, מופעל כמובן לחץ על הבודק לשחררו כבר, ולכן על הבודק לעשות עבודתו נאמנה מבלי לעגל פינות – לעבוד קשה ומהר, אבל נכון ולפי המתודה. הטכניקה שלנו כבודקים היא הדרך שלנו לבצע את המשימה – והתוצאה של עבודתנו תלויה בטכניקה טובה. ניסיון רב ושפשוף בעבודה בגישות מגוונת ומול כלים שונים ותצורות פריסה (מאוד חשובה התשתית הכללית עליה מנהלים את הבדיקות – כלים, הנחיות וכללי אצבע), כשהתהליך האישי מלווה בשיוף, הפקת לקחים ושיכלול תמידי – כל אלו משפרים את הטכניקה שלנו כבודקים ומייעלים אותנו – צריך לשמור על כושר ("להתאמן") כדי להבטיח שנדע לספק תוצאות טובות בעקביות – וצריך להבין ולזכור, שלא נוכל לספק תוצאות אם לא ניצמד לתהליכים ופרוצדורות (כללים המגדירים את דרך ביצועה של משימה מסוימת), ונתעד הכל בכדי שנוכל לאחר מכן לבצע ניתוח תוצאות, ביקורת, הערכה ושיפוט, כדי שתהיה לנו חוות דעת !!! ועוד משהו – אין ספק שאטמות הבדיקות חשוב (כדי שלא נאלץ לבצע אותה משימה שוב ושוב – ולהישחק), אבל הבסיס לפני הפיכת בדיקה לאוטומטית היא לבצע אותה ידנית ולראות בעיניים מה קורה – אחרת לעולם לא נדע אם הסקריפט בכלל מתאים למה שצריך...

כיצד יודעים מתי לסיים את הבדיקות? קשה לייצר תוכנה גדולה ללא באגים, (כפי שיעיד ביל גייטס) וכיוון שרבות מהתוכנות כיום מסובכות ומופעלות בסביבות שונות, בלתי אפשרי פעמים רבות לבצע בדיקה מושלמת.

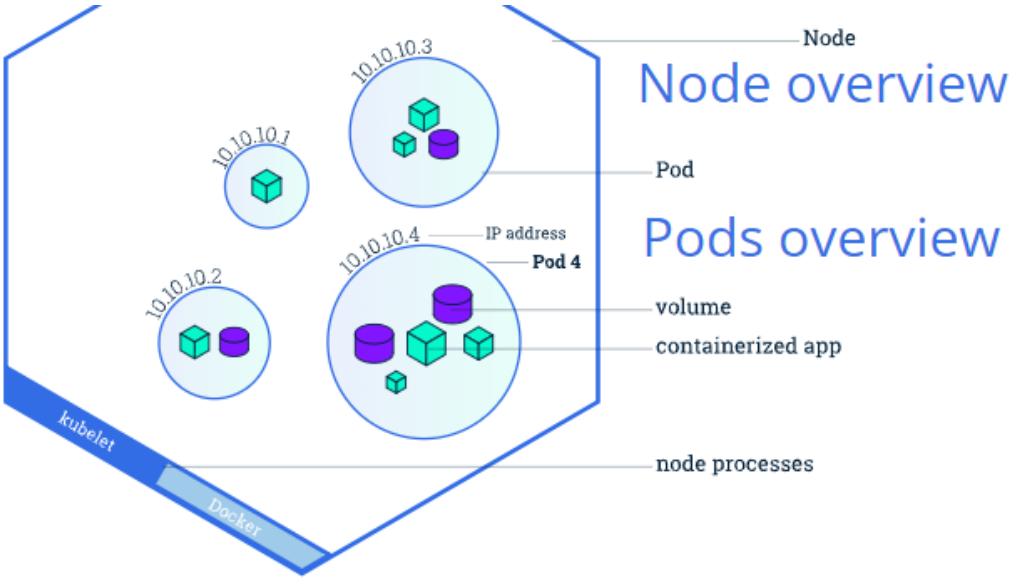
הבדיקה מסתיימת, אם כן, בהתאם לאחת מכמה אפשרויות:

- תאריכי יעד להפצת המוצר או להפסקת הבדיקות
- רמת התקלות ירדה מרמה מסוימת
- מספר וסוג הבדיקות הגיעו לרמה מסוימת שנקבעה קודם לכן
- נגמר תקציב הבדיקה

Kubernetes made Simple

Here are the most important kubectl commands I would recommend a SQA to learn when testing a system using Kubernetes (K8S micro-services system on Cloud):

https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/



Use the following syntax from your terminal: *kubectl [command] [TYPE] [NAME] [flags]*

1. **kubectl get nodes**
   List all nodes in the K8S cluster and provides the essential information about them
2. **kubectl get pods**
   get: List pods running in the cluster and Retrieve info about resources like pods etc.
   kubectl get pods -n my-app
   namespace: Manage namespaces for separating environments and access control
   kubectl get pods --all-namespaces
   events: View events related to Kubernetes resources
   kubectl get events -n my-app
3. **kubectl describe pod <pod-name>**
   describe: Get detailed information about a specific resource pod
   kubectl describe pod my-pod -n my-app
4. **kubectl logs <pod-name>** - Print the logs from a container in a pod
   logs: Access container logs for troubleshooting and debugging
   kubectl logs my-pod -n my-app
5. **kubectl exec <pod-name> -- <cmd>**
   exec: Execute commands inside a running container
   kubectl exec my-pod -n my-app -- sh -c 'curl http://localhost:8080'
6. **kubectl delete pod <pod-name>**
   delete: Remove resources like pods, deployments, or services (delete a pod)
   kubectl delete pod my-pod -n my-app
7. **kubectl rollout status deployment <deployment-name>**
   rollout status: Check the rollout status of a deployment (deployment status check)
   kubectl rollout status deployment my-deployment -n my-app
8. **kubectl cluster-info** - Display cluster info
   cluster-info: Get basic information about the Kubernetes cluster
   kubectl cluster-info

9. **kubectl get services** - List all services in the cluster
10. **kubectl apply -f <file>** - Create resource(s) defined in a file

These provide visibility into resources, applications, logs and connectivity - essentials for testing, debugging and analysis.

Here are some other useful kubectl commands for stopping, starting, and adding resources:

11. **kubectl top nodes** / **top pods**
    Get overall node usage statistics
    kubectl top nodes
    top pods: Monitor resource usage (CPU, memory) of pods.
    kubectl top pods -n my-app
    The -A flag will show pods/usage across all namespaces. Sorting by cpu/memory identifies expensive pods. Node statistics show overall cluster resource utilization.
    Get pods using the most CPU:
    kubectl top pods -A --sort-by=cpu
    Get pods using the most memory:
    kubectl top pods -A --sort-by=memory
12. **kubectl scale deployment <deployment-name> --replicas=<num>**
    scale: Adjust the number of replicas for a deployment - Scale up/down pods' number
    kubectl scale deployment <deployment-name> --replicas=0 (Stopping all replicas)
    kubectl scale deployment my-deployment --replicas=2 -n my-app (Setting 2 replicas)
    To stop an application, scaling its deployment down to 0 pods prevents requests while resources are still defined. Scaling it back up restarts the deployment. Creating from YAML files allows declaring new resources.
13. **kubectl create -f <file>** - Create resource(s) defined in YAML file
14. **kubectl port-forward <pod-name> <local-port>:<pod-port>**
    port-forward: Forward connection to pod port and expose a service port to your local machine for direct testing
    kubectl port-forward svc/my-service 8080:80 -n my-app

Port forwarding allows connecting for debugging/testing. Attaching gets direct access to containers.

Since Kubernetes cluster is running on the cloud, you likely only have remote command line access through one entry point. Here is how you can access and view logs for a specific pod:

1. SSH into the cloud server jump host that provides CLI access to the cluster.
2. Once logged into the jump host, use kubectl get pods to list out pods in the cluster. Identify the name of the target pod.
3. Get the pod's full unique name and namespace:

   kubectl get pods -o wide
4. Access the logs using the pod name and namespace:

   kubectl logs <pod-name> -n <namespace>

   kubectl logs my-app-746d9457f5-trn8m -n production (tail log of a specific pod)

Remember to use **-n** flag to specify namespace, especially when testing multiple microservices. AND, Practice caution when using delete commands, as they are irreversible!

Linux – Useful Linux commands:

1. **ps** - View running processes
2. **top -** Display resource utilization statistics & running processes
3. **df** - See disk space usage
4. **free** - Display memory and swap usage
5. **tail** - Output the last part of log files
6. **grep** - Search files and output matching lines
7. **netstat** - Network connections and stats
8. **curl** - Transfer data to/from a server
9. **wget** - Download files from web servers
10. **kill** - Send signals to terminate processes
11. **sudo** - Run commands with superuser privileges
12. **chmod** - Change file permissions
13. **ssh** - Log into and manage remote servers
14. **git** - Version control system to manage code
15. **vim** / **nano** - Console text editors
16. **tree** - Display directories and subdirectories
17. **alias** - Create shortcuts for long commands
18. **apt** / **yum** - Install and manage software
19. **tar** - Archive multiple files into one
20. **pwd** - which dir am I at

	פקודות בסיסיות
uname -a	נותן את שם המער'
whoami	לבדוק עם איזה יוזר אני במערכת
pwd	מראה באיזו תיקיה אנחנו נמצאים
alias	מראה קיצורים מוגדרים
df -k	מראה ניצול שטח הדיסק
env	מראה משתני סביבה
env | sort	משתני סביבה לפי סדר ABC
history	להראות את כל הפעולות שעשיתי מאז ה-log in
d	תיקיה
-	קובץ
l	לינק
date	מראה תאריך נוכחי
man X	הצג מידע מהי הפונקציה X
cd ~	לוקח לתיקיית היוזר
cd X	לעבור לתיקיה אחרת (X – שם התיקיה)
cd ..	לעלות רמה
ls	להראות רשימה
ls -lrt	להראות רשימה בצורה אנכית, מהסוף, כרונולוגי
Ls -lrta	להראות ברשימה גם קבצים נסתרים
!X	לבצע שוב את פעולה X (מס' הפעולה)
tail X	להראות רק את סוף הקובץ X
head X	להראות רק את תחילת הקובץ X
Ctrl + g	מראה את מספרי השורות שאתה נמצא בהן
Ctrl + L	מחיקת המסך
clear	מחיקת המסך + ממערכת ההפעלה

:v	**לעבור מ Less ל-vi**
mkdir X	יצירת התיקייה X (בתוך התיקיה שנמצאים בה)
\rm -rf x	מחיקת תיקיה X או קובץ X
rm X	מחיקת קובץ X
touch X	יצירת קובץ X בתיקיה הנוכחית
mv X Y	שינוי שם קובץ X לשם חדש Y
mv X Y	העברת הקובץ X לתיקיה y
cp -p X Y	העתקת קובץ X
cp -r X Y	העתקת תיקיה X
grep X *	**חיפוש** מחרוזת X בכל הקבצים בתיקיה זו.
grep X Z	חיפוש בקובץ Z את מחרוזת X
less X	להציג את קובץ X ב-less
	בתוך הקובץ:
less +f x	להציג את הקובץ X מתעדכן אונליין
- x	
Shift f	ממשיך לקבל עדכונים
Ctrl c	עוצר את העדכון
d	קדימה (למטה)
u	אחורה (למעלה)
Shift + G	לסוף הקובץ
1 + Shift g	לתחילת הקובץ
?x	חיפוש מחרוזת x בתוך הקובץ (למעלה
/x	חיפוש מחרוזת (למטה)
n	המשך לחפש למעלה יותר
Shift n	המשך לחפש למטה יותר
:q	יציאה מהקובץ (ב-less)
awk '{print $9}'	** תראה רק את העמודה התשיעית
xargs rm	** מראה את כולם בצורה אופקית, rm - מוחק
ls *.cdr |	תציג את הקבצים שנגמרים ב-cdr, תראה בצורה אופקית
xargs rm	(xargs) ותמחק (rm)
xargs touch	** מראה את כולם בצורה אופקית, touch - שנה לשעה של עכשיו

	פקודות ב VI (visual text editor)
vi X	להיכנס לקובץ X ב-VI
Esc	לעבור למצב command
I	לעבור למצב כתיבה
:q!	לצאת בלי שמירה
:wq	לצאת ולשמור
	במצב command לשוטט בקובץ:
L	ימינה
K	למעלה
J	למטה
H	שמאלה
W	לקפוץ למילה הבאה
B	לקפוץ למילה הקודמת
H	לקפוץ לראש המסך
M	לקפוץ לאמצע המסך
L	לקפוץ לסוף המסך
Ctrl F	המשך למסך הבא
Ctrl B	חזרה למסך הקודם
XG	לעבור לשורה X

u	בטל פעולה אחרונה
U	בטל כל הפעולות בשורה
dd	למחוק שורה
X	למחוק תווים
3x	למחוק 3 תווים
D$	למחוק תווים עד סוף השורה
D0 (zero)	למחוק עד תחילת השורה
dw	למחוק מילה נוכחית
xdd	מחק X שורות
J	אחד שתי שורות
YY	העתקת שורה
P	הדבקת שורה
O / shift O	מוריד שורה למטה (לפני הסמן / אחרי)

_____

Basic commands:

```
man [command]      - display help pages of < command>
sudo [command]     - run < command> in superuser mode
echo -n            - display a line of text
pwd -P             - print current working directory
top                - display current processes, real time monitoring
kill -9            - send signal to process
service [start|stop|restart] - manage or run sysV init script
```

File permission:

```
chmod -c -R        - change file read, write and executable permission
chown -c -R        - change file ownership
```

Network:

```
netstat -r -v      - print network info, routing and connections
telnet             - user interface to the TELNET protocol
ssh -I             - openSSH client
ping -c            - print routing packet trace to host network
```

File Utilities:

```
uniq -c -u         - report or omit repeated lines
head -n            - output the first part of files
diff -q            - file compare, line by line
cat -s             - concatenate files to the standard output
tail -f            - output last part of the file
```

File management:

```
find               - search for a file
ls -a -C -h        - list content of directory
rm -r -f           - remove files and directory
locate -i          - find file, using updatedb
cp -a -R -i        - copy files or directory
du -s              - disk usage
file -b -i         - identify the file type
mv -f -i           - move files or directory
grep, egrep        - print lines matching pattern
```

File compression:

```
tar xvfz          - create or extract .tar or .tgz
gzip, gunzip      - create, extract or view .gz files
rar               - create, extract or view .rar files
```

Directory Utilities:

```
mkdir             - create a directory
rmdir             - remove a directory
```

MORE:

```
find -type f | xargs grep STRING
grep -r "string" /opt
```

```
-----------------------------
Move filles between machines: UNIX
=============================
» SCP:
  -› to same user in other machine:
  user@Machine </dir> : scp <file_name>
<destination_IP:/file's_dir_full_path/>
  -› to other user in other machine:
  user@Machine </dir> : scp <file_name>
<new_user@destination_IP:/file's_dir_full_path/>
» SFTP:
  User@Machine </dir> : sftp <user>@<destination_IP>
  password: <pass>
  sftp> cd </file's_destination_dir_full_path>
  sftp> put <file's_name>
  sftp> quit
» SFTP in Secure CRT in RH example:
  Right click - open SFTP...
  sftp> lcd </file's_dir_full_path/> (on local computer)
  sftp> cd </file's_destination_dir_full_path/>
  sftp> put <file_name>
  Uploading <file_name> to </file's_destination_dir_full_path/><file_name>
  sftp>
» FTP in Secure CRT in Local Lab example:
  START > RUN > cmd (pointing to my Docs&Sett > ftp <destination_IP>
  Name: <user>
  Password: <pass>
  ftp> bin (enter)
  ftp> hash (enter)
  ftp> cd </path/>
  ftp> put <file_name>
  ********************
  ftp> quit

» FTP from Secure CRT sessio to Local Computer (into programFils/SecureCRT
folder)
1. From Secure CRT session -> right-click "Connect SFTP tab" -> Tab
opened...
2. sftp> cd /<full path to file's dir>/
3. sftp> get <file name>
Downloading <file> from <full path>
  100% 1KB      1KB/s 00:00:00     ---> the file was copied to my local
host.
4. Open on LocalHost C:\Program Files\SecureCRT to find File...
```

```
---------------------
Stuff « – ‹ o › – » :
=====================
» Checks for available port / How many connections (sessions) exist on port:
  netstat -na | grep 2222  (2222=port of connection -> if non = not in use -
- and, how many connections on port)
  -› or use:
  telnet 0 2222
  Trying 0.0.0.0...
  telnet: Unable to connect to remote host: Connection refused (port 2222 is
not in use)
» Ping to Host in order to find IP and is Alive:
  ping -a www.walla.co.il
  www.walla.co.il (192.118.82.140) is alive
» Using telnet (connect to port): telnet <your_ip> <the_required_port>
  start > run > cmd ..
  telnet <IP> <PORT> (ip and port your aiming to)
  paste the packet you want to send -› Enter...
» Closing telnet: cnrl+]


-----------------------
Unix Commands for User:
=======================
» Stops actions: cnrl+C
» change user: su -<user_name>
» logout back to the former user: ctrl -d
» Display the history of your actions: h
  Execute action number from the history list: !number
» Delete the current library content: rm -rf *
  Deleting all files in dir begining with x: cd </path/> --> \rm -rf x*
» Copy Here - From - All files -› To copy all files From within a <DIR> X
issue: cp -p </DIR/>* .
» Backup a specific file under another name (p means save the premissions):
  cp -p <file_name> <new_file_name> (if new_file_name is in another place
give the full path)
» Create a file: touch xxx
  Changing file creation time: touch -m MMDDHHMM <file_Name>  ->  touch -m
09011225 xxx
» more <file path + name>: open print of file on visible screen
»
» sort <file>: sorts acording to abc
» Diff between 2 files in diffrent folders:
  diff ./<path1>/<file1> ./<path2>/<file2>
» Find + Replace:
  1. Replace all string values from from within a Folder (dir) to all Files
contained in it (all files at same time):
  as <user> from <dir> issue ->
  bash
  cd </dir_full_path/>
  for i in `ls`;do cp $i ${i}.tmp;sed 's/<current_value>/<new_value>/g' <
${i}.tmp > ${i};rm -f ${i}.tmp;done
  2. Replace all string values from within a File (only one file at a time):
open file in VI and issue ->
  :%s/<current_value>/<new_value>/g
» Change permissions of <file_name> to read write execution (from root
only): chmod 775 <file_name>  (if you also want delete, use 777) ---> for
mor secured mode, use: chmod 664 !!!
» Find and change Line number,Line number: s/ Statement_X /Statement_Y
» Show print lines# 30->31 from log file from anywhewre: sed -n
'<from_line>,<to_line>p' <log_file_full_path>
  in MIM machine/conl: sed -n '30,31p' $LOG_DIR/mim2/mim2.log
» Copy Paste Of "Code Section" To Tmp Directory.txt examples:
  sed -n '/13 Dec 2006 09:34:05/,/13 Dec 2006 09:34:44/ p'
$LOG_DIR/mim4/mim4.log > /tmp/MIM4_Exception_
  sed -n '/9284/,/7486/ p' $LOG_DIR/mim_portal_gw7/mim_portal_gw.log >
/tmp/PORTAL7
  sed -n '99322,99509 p' $LOG_DIR/mim_sap_gw7/mim_sap_gw.log > /tmp/Sap7
```

```
» System file that include among others alias commands:  .cshrc
  as root  ->  cd ~  ->  ls -la  ->  less .cshrc  ->  Edit + save change
that you made in the file  ->  source .cshrc
» To find a core file - issue: find . -name "*core*"
» Pstack: pstack <core_file>
» tar+gzip:
       gzip FOLDER.tar
       gunzip FOLDER
       tar -vcf FOLDER.tar FILE FILE FILE
       tar -xvf FOLDER
       gtar -xzf FileName.tar.gz

» chown user:group dir (eg. chown odigo314:odigo CONFIG_FILES-
BackUp_3.1.4_3)

» Edit Aliases:
       As user perform: cd ~
       vi .login -> then issue: source .login

» System Aliases:
       As user perform: alias (to see alias on machine)


----------------
VI - edit files:
================
» To open <file> in "VI edit mode" -> issue:   vi <file>
» To get Into "Row Number view" in VI mode:    :set nu
» Into Writing mode from infront of cursor:    i
» Into Writing mode from begining of line:     shift+i
» Into Writing mode from after the cursor:     a
» Into Writing mode from the end of the line:  shift+a
» Line down after cursor & Into Writing mode:  o
» Line down before cursor & Into Writing mode: shift+o
» Out without saving:                          :q!
» Out with saving:                      :wq!
» Into Command mode (no writing):       Esc
  At command mode you can perform ->
» l          Navigate Right
» k          Navigate Up
» j          Navigate Down
» h          Navigate Left
» w          Navigate to Next Word
» b          Navigate to Prev Word
» shift+h    Navigate to Screen Top
» shift+m    Navigate to Screen Middle
» shift+l    Navigate to Screen Bottom
» ctrl+f     Navigate to Next Screen
» ctrl+b     Navigate to Prev Screen
» X shift+g  Navigate to line X
» u          Undo the last action
» shift+u    Undo all actions in line
» dd         Delete a line
» Xdd        Delete X lines
» x          Delete 1 character
» 3x         Delete 3 characters
» d$         Delete all characters till the end of the line
» d0 (zero)  Delete all characters till the line begins
» dw         Delete current word
» shift+j    Merge 2 lines
» yy         Copy a line
» p          Paste a line
» .          Repeat the last action
»
```

```
-------------------------
Unix Commands for System:
=========================
» To get information about disk space of Directory (as root): du -s -h
</path/dir>
» To get information about disk space: df -k , du -sh (xx -xh -> the h
letter display the size of the file)
» To view hard disk space and partitions:   fdisk -l
» To get information about process (UID/PID/CMD): ps -ef
  ps -ef | grep <strin_within_process_path_name> (for a specific process)
» Process Monitoring:
  /usr/ucb/ps uax | head                              - display most active
processes
  ps -eo pid,vsz,args | sort +1n                      - Display processes
with the highest MEM usage
  ps -eo pid,pcpu,args | sort +1n                     - Display processes
with the highest CPU utilization
  ps -ef | grep <name> | grep -v grep | awk '{print $2}'    - find the PID
of process by its name. run from telnet or SSH not console!
» Display CPU Level (PID USERNAME CPU PROCESS): prstat
» Kill processes: Kill -9 pid
» Display the current machine: ifconfig -a
» ????: netstat  //  netstat -an
» Verify it doesn't listen to port:   netstat -na | grep <PORT>
» See ALL open ports: netstat -tulanp
» Mount cdrom: mount /dev/hdc -t iso9660 -r /cdrom
» What is currently running on machine: list-running -local
» ssh <IP> - enter machine
» ifconfig - find IPs in machine
» hostname


---------
More JUNK
=========
cat /proc/cpuinfo          - CPU information
cat /proc/meminfo          - memory information
cat /proc/version          - linux distro version
cat /etc/redhat-release          - red-hat version
chmod -R 777 *                   - recursively change the permissions of all
files and directories
coreadm                          - specify the name and location of core
files produced by abnormally-terminating processes
cp -r                      - cp will recursively copy the directory and all
its subdirs and files.
clear                      - clear the screen
dmesg                      - print out bootup messages
dmidecode                  - reports info about your system's hardware as
described in your BIOS (Model, S/N etc')
date                       - display current date and time
date <MM:DD:YYYY>          - change date time to MM:DD:YYYY (date 1234.56
changes to 12:34:56)
date -s <hh:mm:ss>         - change time to hh:mm:ss
df -ak                     - Report filesystem disk space usage in kilobytes
du -sh /dir/               - check the size of directory in human readable
format. try also: du -sk /dir
eject cdrom                - open the CDROM drive
ifconfig -a                - display current configuration of all NICs
ifconfig -a | grep net           - display IP addresses of NICs in short
format
insmod                     - install loadable kernel module
gtar zxvf <filename>.tgz   - extracts a tgz package
gunzip <filename>.tgz            - extract tgz to a tar package
gzip -c9 core > core.gz &  - compress core to file core.gz
ifconfig -a                - display current  configuration of all  network
interface
isainfo -b                 - check size of instruction set
kill %1                          - kill suspended processes number 1
```

```
last                        - see last logged in users (pseudo user reboot
logs in each time the system is rebooted)
less -i                           - view a file in case insensitive mode
ls -lt                      - list files and sort by last modified date
ls -ld                      - lists only directory names (not  their
contents)
ls -ld <pattern>            - lists only directory names (not  their
contents). Example: ls -ld CP*
ls -a | grep '^\.'          - list only hidden files. To list by last modified
(ascending order): ls -ltar | awk '{print $9}' | grep '^\.'
lsmod                       - list loaded modules(drivers).
modinfo                           - display information about loaded kernel
modules(drivers)
netconf                           - GUI to configure NICs
netstat -nr                 - see routing table
netstat -an | grep LISTEN   - see ALL LISTENING ports
ndd                         - get and set driver configuration parameters
pkgrm                       - uninstall packages
pkgadd -d .                 - install packages located in current directory
prtconf -v | grep Memory    - display amount of memory
prstat                      - report active process statistics
prtdiag                           - display no. of CPUs and system
diagnostics.
psrinfo -v                  - processor type and speed (Mhz/Ghz)
rdate B                           - synchronize current machine's clock with
machine B's clock (both machines should be Solaris)
rm -rf                      - remove a whole directory tree without warnings
(BE CAREFUL)
rm ./-v                           - to delete "-v" (a hard to remove file),
notice the "./" before the filename.
rmmod                       - unload loadable modules from the running kernel
rpm -ivv <package_name>           - debug installation of RPM
rpm -qa CP*                 - see which CP packages are installed
rpm -ql <package_name>            - list files associated with package
rpm -e <package_name>             - uninstalls the package
rpm -i <package_name>             - installs the package
shutdown now                - shutdown machine
snoop                       - capture and inspect network packets
stat /path                  - display file or filesystem status (view access,
modify and change time).
strace -s 1024 -p <pid>           - trace system calls and signals (length of
strings=1024) of <pid>
strings <filename>          - find printable strings in an object or binary
file
tar xvf <filename>          - extracts a tar package
tar ztf <filename>.tgz            - list contents of tar/tgz package.
tcpdump                           - dump traffic on a network
top                         - display top CPU processes
truss -p <pid>                    - trace system calls and signals of <pid>
truss /bin/ls               - trace system calls and signals of /bin/ls
uname -a                    - display name of current system
w                           - display information about currently logged-in
users
which <filename>            - locate a command; display its pathname or alias
(to determine which executable is used!)
who -r                      - determine current runlevel
```

```
------------
Grep search: UNIX
============
» grep "_dump" $CONFIG_FILES_DIR/*params
» egrep -i TERM * (non key sensitive) | grep another term * | grep -v #
(narrow the result not to include comments)
» mim_sap_gw_server.params |egrep -i appaut |egrep -i enab
» egrep -i GroupId * $CONFIG_FILES_DIR/repository.params
» run-omap mim -id 1 -exec 'query_user 1234' | grep -i CONTACT_ID
```

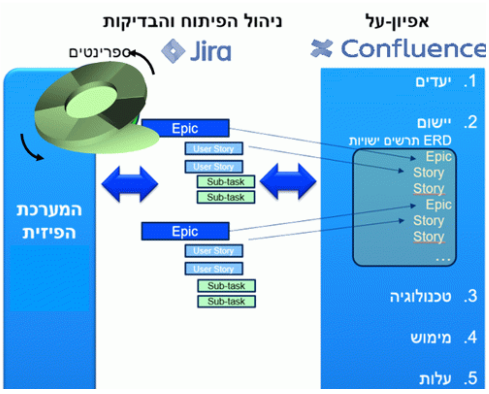
```
» egrep on tow strings:
  grep 9253243947 mim1.log.[1-6] | egrep Opcode=700\|Opcode=701
» Search from outside Config Directory for "Term" in "Path":
  grep Term $CONFIG_FILES_DIR/*
» Search inside Config Directory for "Term" with no key sensativity:
  egrep -i term *
» Search from outside Config Directory for "Term" in the repository.params
file -with no key sensativity:
  egrep -i TERM * $CONFIG_FILES_DIR/repository.params
» Find all params containing "DB" in a file:
  cd $CONFIG_FILES_DIR/
  less repository.params | grep DB
» Search outside Config Directory for "Term" in repository.params file.
  less $CONFIG_FILES_DIR/repository.params | grep Term
  less $CONFIG_FILES_DIR/repository.params | grep YAHOO | grep IP
» Go to Config -› cd $CONFIG_FILES_DIR/
  grep -i minimumIntervalBetweenPresenceNotifications *
» Search String (not Key-sensitive)
  less $CONFIG_FILES_DIR/mim_server.params  | egrep -i minimum | egrep -i
interval |   egrep -i notification
  cd $CONFIG_FILES_DIR/
» In MIM, SAP, PORTAL » Enter Config
  [odigo@OIM-QA3 ~/config_files]$ grep "Y! Mobile Messenger" *
» Search for string "X" (the " is because ob spases):
  User@Machine </dir> : egrep -i service * | grep Ip   (try to find it)
  User@Machine </dir> : less mim_portal_gw_server.params | grep
ExternalServices.AIM | grep -v #   (find it)
  User@Machine </dir> : less mim_portal_gw_server.params.good |grep
ExternalServices.ICQ.IP
» Search in another way...
  User@Machine </dir> : grep -i private mim_portal_gw_server.params
  ExternalServices.AIM.PrivateClientKey1 = AOL Instant Messenger


-------------
Find in file: UNIX
=============
» Quick Locate -> As root: locate <part the of file's name> (to file all
files containing it, before issue: updatedb -> to update the index)

» To find a string in a file:
  grep "kkkkkk" `find .`
» find . -name "*core*" or find . -name "Mylog.txt" --- or: find -name
"*iddleware.confi*"
» Find a string recursively in directory(s)  (BEST version i've found):
  grep -in '<string>' `find . -type f` | grep -v "^Binary file"
  find . -type f -name <file>                        - run from root
directory to search for <file>
  find . -type f -exec grep '<string>' /dev/null {} +- find a string
recursively in directory(s)
  find . -type f -print | xargs grep -i <string>            - find a string
recursively in directory(s)
  find . -exec egrep -li 'clinton_ext' '{}' \;       - only list all
matching files recursively
  find . -depth -name <file>                         - run from root
directory to search for <file>
» Recursive grep (in this example we look for “ssl_connect”):
  find . -name \* -exec grep ssl_connect \{} \; -print


.
```

התיכנון גמיש: AGIL

יישום עקרונות האג'יל באמצעות מתודולוגית מפת"ח האג'ילי מחייב שימוש בכלים ממחושבים לניהול תהליכי הפיתוח.

קיט זה, מתמקד בתהליכים הבסיסיים, Best practices , ליישום MethodAgile באמצעות כלי Atlassian וכלים אחרים

Jira Software - ניהול פרויקטים, משימות חותהליכי עבודה



כלים

JIRA SOFTWARE	ניהול פרויקטים, משימות ותהליכי העבודה	
Confluence	ניהול, תיעוד ושיתוף ידע באמצעות מערכת	
Service Desk	פתרון כולל לניהול השירות	
eazyBI	מאפשר יצירת דוחות, גרפים ו-Dashboards בצורה קלה	
Bitbucket	Bitbucket gives teams one place to plan projects, collaborate on code, test, and deploy	
Portfolio for Jira	Helps you build a plan, see the big picture, track progress, and easily share with stakeholders.	

מצגות

https://www.methoda.cloud/Content/pages/kit_tools/H_home-map.asp

שלבים השונים במסלול חיי התוכנה

- הגדרת הדרישות - Requirements
- תכנון - Design
- תכנות - Coding
- בדיקות מפתחים - Developer Tests
- בדיקות - QC
- תיעוד – Documentation

DB & SQL

```
---------
DataBase:
=========
» Our Oracle  DB:
  enter machine as oracle: su - oracle
  open SQLplus: sqlplus <Schema_User>/<Password>@<Service_Name>
  type: exit (to exit DB)
  Out of the SQL+: ps -ef | grep pmon » show which DB schemas are up.
  Out of the SQL+: echo $ORACLE_SID » show which DB you are connected to.

» To Switch Userfrom within SQL+: conn
<Schema_User>/<Password>@<Service_Name>
» To find out which tables in Schema:
  SQL> select table_name from tabs;
» To see all columns in a table issue:
  SQL> describe mim_user_agent;
» To Select column-a's value of record in case column-b equals 0:
  SQL> select <column-a> from <table> where <column-b>=0;
» Simple queries:
  select * from <table> where <column> like '%imor%';
» Stopping the connection  and Reconecting the Server and DB:
  Enter oracle
   [root@T1000U /]# su - oracle (password = oracle)
                   ---------------------------------
  Change variable to the required DB schena
   -bash-3.00$            export ORACLE_SID=ODSAPP
                   ------------------------
  Enter as sysdba (he is the only one authorized)
   -bash-3.00$            sqlplus '/as sysdba'
                   --------------------
  Verify you are in the right DB = NAME - ODSAPP
   SQL>            select name from v$database;
                   ---------------------------
  Disconnect / Connect commands:
   SQL>            shutdown immediate    -or-
   SQL>            startup
  To exit DB --->
   SQL>                   exit
» DB Listener restart - if NO Veritas:
  If DB is UP and it is not connecting -
  Check the Listener:
  SQL> exit
  -bash-3.00$ lsnrctl
  LSNRCTL> status
  LSNRCTL> start
  LSNRCTL> exit
» Back Up table aaa:
  SQL> create table bbb as select * aaa ;
» Copy data from one DB to another - by Tables:
  -> from DB root: su - oracle
  SQL> conn system/j1u9cV783
  Connected.
  SQL> delete from  MIM314_CLP.mim_user_agent ;
  1 row deleted.
  SQL>  insert into MIM314_CLP.mim_user_agent select * from
MIM314.mim_user_agent ;
  162 rows created.
  SQL> commit;
  Commit complete.
  * if you are already in sqlplus you can issue only connect (insread of out
+ in again as a new user):
  conn system/j1u9cV783
» drop user <SCHEMA> cascade ;
» Delete all data from tabe and insert data from another schema table:
  SQL> delete from <schema-A>.<table> ;
```

```
  SQL> insert into <schema-A>.<table> select * <schema-B>.<table> ;
  SQL> commit;
» OMS DB Jobs -
  Connect to the  OMS server DB: sqlplus oms314/oms314@ods
  In order to see all the Jobs:
       select id,status,name,last_seq,base_table from oms_summaries;
  In order to issue job 12 for example:
       exec summary_manager.run_summary(12);
» More Examples:
  SQL> select max(user_agent_id) from MIM_USER_AGENT;
  SQL> select USER_AGENT_SEQ.nextval from dual;
  SQL> alter sequence USER_AGENT_SEQ increment by 20000 ;

» Drop Sequence:
  drop sequence UNIQUE_ID_SEQ;

  CREATE SEQUENCE UNIQUE_ID_SEQ START WITH 1
  INCREMENT BY 1
  MINVALUE 1
  MAXVALUE 100000000
  NOCACHE
  CYCLE NOORDER ;

» Joins of 2 Tables
Table X:            Table Y:
Name | ID |         Name | Job
 a   | 1  |          b   | M
 b   | 2  |          c   | N
 c   | 3  |          d   | M

Select X.Name, X.ID, Y.Job From X join Y on X.Name = Y.Name Where Y.Job = M;

Table result:
X.Name | X.ID | Y.Job
 b     | 2    | M
=====================
  The action How to run in sqlplus
  --------------------------------
  0    select all of the tables of this schema / select all of the views /
select all of the synonym ->
       select table_name from tabs ;
       select view_name from user_views;
       select synonym_name from user_synonyms;
  1    see the table columns ->
       describe <TABLE_NAME> ;
  2    see all the values in one of the columns ->
       select <COLUMN_NAME>, <COLUMN_NAME> from <TABLE_NAME> ;
  3    see values of columns where value of column x = 4 and column Y=WORD ->
       select <COLUMN_NAME>, <COLUMN_NAME> from <TABLE_NAME> where
<COLUMN_NAME>=4 and where <COLUMN_NAME>='WORD' ;
  4    see values of columns where value of column x has name that contains
the string "ggg" ->
       select <COLUMN_NAME>, <COLUMN_NAME> from <TABLE_NAME> where
<COLUMN_NAME> like '%ggg%' ;
  5    Update the table -> where COLUMN_NAME=A change COLUMN_NAME to B ->
       Update <TABLE_NAME> set <COLUMN_NAME>='B' where <COLUMN_NAME>='A' ;
  6    see the table sorted by column A ->
       select <COLUMN_NAME>, <COLUMN_NAME> from <TABLE_NAME> ordered by
<COLUMN_NAME> ;
  7    add a raw ->
       Insert into <TABLE_NAME>
('<COLUMN_NAME>','<COLUMN_NAME>','<COLUMN_NAME>') values
('<value>','<value>','<value>') ;
  8    Delete a raw ->
       delete from TABLE_NAME where…
  9    copy a table ->
       create table mim_user_agent_new as select * from mim_user_agent;
  10   delete a table ->
```

```
      drop table mim_user_agent_new
  11  select the max value of column X=TIMESTAMP in table=oms_cdr_msg ->
      select max(TIMESTAMP) from oms_cdr_msg ;
  12  check how many rows are there in the table=mim_string_map ->
      select count(*) from mim_string_map;
  13  Truncate table ->
      truncate table OMS_CDR_TRANSACTION ;
  14  commit; -or to undo -> rollback;
  15  Select and change STRING_VALUE column hex to text ->
      select STRING_CODE,PORTAL_ID,CLIENT_LANGUAGE, USER_AGENT_ID,
utl_raw.cast_to_varchar2(STRING_VALUE) from mim_string_map where
STRING_CODE=108;
  16  run a file that was saved earlier ->
      sqlplus mim/mim@odsapp @upd.sql
  17  Select and show timestamp as date ->
      select secs2date(timestamp) from…
  18  Show time, not only date ->
      alter session set nls_date_format='dd_mm_yy hh24:mi:ss';
  19  set the view of a table (--- --- ---) ->
      set pages 10
      set lines 444
      col <column_name> for a25 (if chars)
      col <column_name> for 99999 (if num)
  20  repeat last action: /
      see last action: l
  21  desc <table> = describe the table's columns
```

– END –